

11-06-00

A

11/03/00



36961 U.S.

**IN THE UNITED STATES PATENT AND TRADEMARK OFFICE**

**CERTIFICATE OF EXPRESS MAILING**

I hereby certify that this paper and the documents and/or fees referred to as  
therein are being deposited with the United States Postal Service  
on November 3, 2000 in an envelope as "Express Mail Post Office to  
Addressee" service under 37 CFR §1.10, Mailing Label Number  
**EL580854097US**, addressed to the Assistant Commissioner for Patents,  
Washington, DC 20231.

*Mandy Franklin*  
Mandy Franklin

Attorney Docket No.: DANAP004

First Named Inventor: SRINIVAS



1c825 U.S. PTO

09/706296

11/03/00

**UTILITY PATENT APPLICATION TRANSMITTAL (37 CFR. § 1.53(b))**

(Regular application claiming priority of a provisional)

Assistant Commissioner for Patents  
Box Patent Application  
Washington, DC 20231

Sir: This is a request for filing a patent application under 37 CFR. § 1.53(b) in the name of inventors:  
**Sampath Srinivas, Theron Tock**

For: **DYNAMIC TOOLBAR FOR MARKUP LANGUAGE DOCUMENT**

Priority of U.S. Provisional Application No. 60/235,513 filed on September 26, 2000 is claimed  
under 35 U.S.C. § 119(e).

**Application Elements:**

- ☒ **50** Pages of Specification, Claims and Abstract
- ☒ **19** Sheets of informal Drawings
- ☒ Combined Declaration and Power of Attorney
- ☐ Separate Declaration

**Accompanying Application Parts:**

- ☒ Assignment and Assignment Recordation Cover Sheet (recording fee of \$40.00 enclosed)
- ☐ Separate Power of Attorney by Assignee
- ☐ 37 CFR 3.73(b) Statement by Assignee
- ☐ Information Disclosure Statement with Form PTO-1449
  - ☐ Copies of IDS Citations
- ☐ Preliminary Amendment
- ☒ Return Receipt Postcard
- ☒ Small Entity Statement(s)

☐ Other:

Fee Calculation (37 CFR § 1.16)

	(Col. 1) Total Claims		(Col. 2) Claims	(Col. 3) Present Extra	Rate	Additional Fee
TOTAL	21	MINUS	20	= 01	x 18	18.00
INDEP.	05	MINUS	3	= 02	x 80	160.00
[ ] First presentation of multiple dependent claim					\$270	
Basic Filing Fee under 37 C.F.R. §1.16(a)					\$710	710.00
TOTAL						888.00
SMALL ENTITY 50% FILING FEE REDUCTION (if applicable)						444.00

☒ Check No. 9468 in the amount of \$484.00 is enclosed.

☒ The Commissioner is authorized to charge any fees beyond the amount enclosed which may be required, or to credit any overpayment, to Deposit Account No. 500388 (Order No. DANAP004).

General Authorization for Petition for Extension of Time (37 CFR §1.136)

☒ Applicants hereby make and generally authorize any Petitions for Extensions of Time as may be needed for any subsequent filings. The Commissioner is also authorized to charge any extension fees under 37 CFR §1.17 as may be needed to Deposit Account No. 500388 (Order No. DANAP004).

☒ Please send correspondence to the following address:

**Customer Number 022434**  
**BEYER WEAVER & THOMAS, LLP**  
P.O. Box 778  
Berkeley, CA 94704-0778  
Telephone (650) 961-8300  
Fax (650) 961-8301



Date: 11/3/00

*C. Douglass Thomas*

**C. Douglass Thomas**  
Registration No. 32,947

Applicant/Patentee: Srinivas et al.  
Application or Patent No. Not Yet Assigned Atty Docket # DANAP004  
Filed or Issued: Herewith

VERIFIED STATEMENT (DECLARATION) CLAIMING SMALL ENTITY STATUS  
37 CFR 1.9(f) and 1.27(c)--SMALL BUSINESS CONCERN

I hereby declare that I am

- ☐ the owner of the small business concern identified below;  
☐ an official empowered to act on behalf of the small business concern identified below;

NAME OF CONCERN: Danastreet Internet, Inc.  
ADDRESS: 3443 Georgetown Place, Santa Clara, CA 95051

I hereby declare that the above identified small business concern qualifies as a small business concern as defined in 13 CFR 121.3-18, and reproduced in 37 CFR 1.9(d), for purposes of paying reduced fees under 41(a) and (b) of Title 35, U.S. Code, in that the number of employees of the concern, including those of its affiliates, does not exceed 500 persons. For purposes of this statement, (1) the number of employees of the business concern is the average over the previous fiscal year of the concern of the persons employed on a full-time, part-time or temporary basis during each of the pay periods of the fiscal year, and (2) concerns are affiliates of each other when either, directly or indirectly, one concern controls or has the power to control the other, or a third party or parties controls or has the power to control both.

I hereby declare that rights under contract or law have been conveyed to and remain with the small business concern identified above with regard to the invention entitled: DYNAMIC TOOLBAR FOR MARKUP LANGUAGE DOCUMENT, by inventor(s) Sampath Srinivas and Theron Tock, described in


- ☒ the specification filed herewith.  
☐ Application No. \_\_\_\_\_ filed \_\_\_\_\_  
☐ patent # \_\_\_\_\_ issued \_\_\_\_\_

If the rights held by the above-identified small business concern are not exclusive, each individual, concern or organization having rights to the invention is listed below\* and no rights to the invention are held by any person, other than the inventor, who could not qualify as a small business concern under 37 CFR 1.9(d) or by any concern which would not qualify as a small business concern under 37 CFR 1.9(d) or a nonprofit organization under 37 CFR 1.9(e).  
\*Note: separate verified statements are required from each named person, concern or organization having rights to the invention averring to their status as small entities. (37 CFR 1.27)

Name: \_\_\_\_\_  
Address: \_\_\_\_\_  
☐ individual ☐ small business concern ☐ nonprofit organization

I acknowledge the duty to file, in this application or patent, notification of any change in status resulting in loss of entitlement to small entity status prior to paying, or at the time of paying, the earliest of the issue fee or any maintenance fee due after the date on which status as a small entity is no longer appropriate. (37 CFR 1.28(b)).

I hereby declare that all statements made herein of my own knowledge are true and that all statements made on information and belief are believed to be true; and further, that these statements were made with the knowledge that willful false statements and the like so made are punishable by fine or imprisonment, or both, under 1001 of Title 18 of the U.S. Code, and that such willful false statements may jeopardize the validity of the application, any patent issuing thereon, or any patent to which this verified statement is directed.

NAME OF PERSON SIGNING: Sampath Srinivas  
TITLE IN ORGANIZATION: CEO  
SIGNATURE  DATE 03 NOV 2000

# PATENT APPLICATION

## DYNAMIC TOOLBAR FOR MARKUP LANGUAGE DOCUMENT

Inventor(s): 1. Sampath Srinivas  
3443 Georgetown Place  
Santa Clara, CA 95051  
Citizenship: USA

2. Theron Tock  
620 Palo Alto Ave.  
Mountain View, CA 94041  
Citizenship: USA

Assignee: Danastreet Internet, Inc.

# DYNAMIC TOOLBAR FOR MARKUP LANGUAGE DOCUMENT

## CROSS-REFERENCE TO RELATED APPLICATION

5 This application claims the benefit of U.S. Provisional Patent Application No. 60/235,513, filed September 26, 2000, and entitled "ENHANCED BROWSING ENVIRONMENT", and which is hereby incorporated by reference herein. This application is also related to concurrently filed U.S. Patent Application Nos. X1, X2, and X3.

## BACKGROUND OF THE INVENTION

### **1. Field of the Invention**

10 The present invention relates to client-server computing and, more particularly, to client-server computing for accessing resources over a network.

### **2. Description of the Related Art**

15 Network browsers (browser applications), such as Netscape Navigator or Microsoft Explorer, allow users of client machines to request and retrieve resources from remotely located server machines via the Internet. These network browsers can display or render HyperText Markup Language (HTML) documents provided by the remotely located server machines. Additionally, browsers are able to execute script programs embedded in the HTML documents to provide some local functionality. More recent network browsers support a searchable history list, a favorites or bookmarks list, etc.

25 Although traditional network browsers are very useful, there is a need to provide users of network browsers with access to increased functionality and services. Generally, increased functionality and services can be provided to network browsers by (i) functionality built into network browsers, (ii) services provided by plug-in software, or (iii) web based services.

Functionality can be built into network browsers. However, because network browsers are designed for general, local use, only general functions get incorporated into network browsers. Examples of built-in functionality include a searchable history list, a favorites or bookmarks list, etc. which are provided in more recent network browsers.

Various specific browser enhancements can be provided (typically, via third parties) by software plug-ins that modify the network browser. As such, to make use of these enhancements, special purpose plug-in software needs to be downloaded to and installed on a client machine. Having to download software to obtain an enhancement is burdensome and often discourages users from obtaining the enhancement. Examples of plug-ins include LiquidAudio Media Player which allows audio sound files to be played, ThirdVoice.com which facilitates a browser companion service that allows users to add comments to any webpage, etc.

There is a growing trend to move services and functionality to the Internet (World Wide Web) and to provide access to these services through a simple network browser. As such, building functionality into network browsers or providing plug-ins are not desirable approaches. Although web-based services are desirable for this trend, various companies have developed their own server-side architecture to enable their web-based services. Examples of some web based services include: anonymizer.com which provides anonymity by routing request through their website; netmind.com which allows monitoring for changes to web pages; and desktop.com which provides a web desktop (a portable web space). While these websites may be able to normally provide support for their services, they do so with a special purpose server-side design and do not provide a consistent or generally useful platform for supporting a wide range of services.

Thus, there is a need for providing a web-based platform that is capable of supporting a wide range of services.

## SUMMARY OF THE INVENTION

The invention pertains to a toolbar that is provided or inserted in a markup language document so as to facilitate features or functionality provided by a server. The toolbar is able to determine whether the toolbar should be displayed as part of the markup language page being displayed. In one embodiment the server is an intermediary server.

The invention can be implemented in numerous ways, including as a system, method, device, and a computer readable medium. Several embodiments of the invention are discussed below.

As a method for inserting a toolbar into a webpage at a server machine, one embodiment of the invention includes at least the acts of: receiving a webpage at the server machine to be delivered to a client machine; inserting the toolbar into the webpage at the server machine, the toolbar including at least one link to a resource and an executable script; and delivering the webpage to the client machine.

As a method for modifying a markup language page to include a dynamically determinable toolbar according to one embodiment of the invention, the method includes at least the acts of dynamically determining whether the dynamically determinable toolbar should be displayed as part of the markup language page when said markup language page is being displayed.

As a computer readable medium including at least computer program code for determining whether a toolbar should be display in one or more frames of a webpage, said computer readable medium includes at least: computer program code for determining whether the toolbar is within the one or more frames of the webpage; computer program code for determining the size of the one or more frames of the webpage; and computer program code for displaying the toolbar in each of the one or more frames of the webpage in which the size of the one or more frames exceeds a threshold size and suppressing displaying the toolbar in each of the one or more frames of the webpage in which the size of the one or more frames does not exceed the threshold size.

As a computer readable medium including at least computer program code for inserting a toolbar into a webpage at a server machine, said computer readable medium includes at least: computer program code for receiving a webpage at the server machine to be delivered to a client machine; computer  
5 program code for inserting the toolbar into the webpage at the server machine, the toolbar including at least one link to a resource and an executable script; and computer program code for delivering the webpage to the client machine.

As a dynamically determinable toolbar, one embodiment of the invention provides the dynamically determinable toolbar in a markup language page, the  
10 dynamically determinable toolbar operates when the markup language page is being displayed to dynamically determine whether said dynamically determinable toolbar should be displayed as part of the markup language page.

Other aspects and advantages of the invention will become apparent from the following detailed description taken in conjunction with the  
15 accompanying drawings which illustrate, by way of example, the principles of the invention.

### **BRIEF DESCRIPTION OF THE DRAWINGS**

The invention will be readily understood by the following detailed  
20 description in conjunction with the accompanying drawings, wherein like reference numerals designate like structural elements, and in which:

FIG. 1 is a block diagram of an information retrieval system according to one embodiment of the invention;

FIG. 2A is a block diagram of an intermediary server according to one  
25 embodiment of the invention;

FIGs. 2B and 2C illustrate an example of a computer system that may be used in accordance with the invention;

FIG. 3 is a flow diagram of intermediary request processing according to one embodiment of the invention;



FIGs. 4A and 4B are flow diagrams of client processing according to one embodiment of the invention;

FIGs. 5A - 5D are flow diagrams of intermediary processing according to one embodiment of the invention;

5 FIG. 6 illustrates a diagram of an information retrieval system according to one embodiment of the invention;

FIG. 7 is a diagram of application plug-in processing according to one embodiment of the invention;

10 FIG. 8 illustrates a database and a file server according to one embodiment of the invention;

FIG. 9A illustrates a toolbar activation process according to one embodiment of the invention;

FIG. 9B is a representative toolbar that can be inserted into a webpage in accordance with one aspect of the invention;

15 FIG. 10 is a flow diagram of URL modification processing according to one embodiment of the invention;

FIG. 11 is a flow diagram of a script modification processing according to one embodiment of the invention; and

20 FIGs. 12A and 12B are flow diagrams of a script modification processing according to another embodiment of the invention.

### **DETAILED DESCRIPTION OF THE INVENTION**

25 The invention pertains to a toolbar that is provided or inserted in a markup language document so as to facilitate features or functionality provided by a server. The toolbar is able to determine whether the toolbar should be displayed as part of the markup language page being displayed. In one embodiment the server is an intermediary server.

Embodiments of this aspect of the invention are discussed below with reference to FIGs. 1 – 12B. However, those skilled in the art will readily

appreciate that the detailed description given herein with respect to these figures is for explanatory purposes as the invention extends beyond these limited embodiments.

FIG. 1 is a block diagram of an information retrieval system 100 according to one embodiment of the invention. The information retrieval system 100 includes a network 102, client machines 104 and 106, an intermediary server 108, and remote servers 110 and 112. The network 102 serves as a communication medium through which the client machines 104 and 106, the intermediary server 108 and the remote servers 110 and 112 can communicate. The network 102 is, for example, a data network which can include the Internet, a wide area network, or a local area network. The Internet refers to a global network of interconnected computers.

According to the invention, requests for content residing on the remote servers 110 and 112 can be received from the client machines 104 and 106. As used herein "content" is any information or resource that can be stored on a server and retrieved by a client. Typically, the content is embodied as an electronic file and contains text and/or images. Often the client machines 104 and 106 operate browser applications that facilitate requesting and retrieval of content over the network 102. In such case, the content is often returned to the browser application as a browser viewable document (e.g., markup language document, webpage, etc.) so that the browser application can display the same. Instead of the client machines 104 and 106 directly accessing the remote servers 110 and 112 through the network 102, the client machines 104 and 106 communicate with an intermediary server 108. The intermediary server 108 then, in turn, accesses the remote servers 110 and 112 on behalf of the client machines 104 and 106. Once the intermediary server 108 obtains the requested content from the remote servers 110 and 112, the intermediary server 108 can directly return the requested content to the client machines 104 and 106 or can first modify the requested content and then deliver it to the client machines 104 and 106.

The modification to the requested content by the intermediary server 108 can take a variety of forms. As one example, the intermediary server 108 can insert a toolbar into the requested content before delivery to the client machines 104 and 106. As another example, the intermediary server 108 can alter the hyperlinks within the requested content so as to point to an intermediary server (e.g., the intermediary server 108). Various other tasks can be performed on the requested content by the intermediary server 108.

Additionally, the information retrieval system 100 can also support centralized storage at the intermediary server 108 of server stored information as well as previously requested content. The server stored information is often referred to as "cookies", though cookies are conventionally stored on client machines. The centralized storage of the previously requested content at the intermediary server 108 can provide a history of content previously requested and returned to client machines. The history can then be later retraced, searched, etc. to locate previously retrieved content.

Although the information retrieval system 100 illustrated in FIG. 1 depicts only a pair of client machines, a pair of remote servers and a single intermediary server, it should be understood that the information retrieval system 100 can support many client machines and many server machines. It should also be understood that the information retrieval system 100 can also support multiple intermediary servers.

FIG. 2A is a block diagram of an intermediary server 200 according to one embodiment of the invention. The intermediary server 200 is, for example, suitable for use as the intermediary server 108 illustrated in FIG. 1.

The intermediary server 200 includes various processing modules typically implemented by computer program code executed by a processing device utilized by the intermediary server. More particularly, the processing modules of the intermediary server 200 include a web server 202 and a HyperText Transfer Protocol (HTTP) handler 204. The web server 202 couples to client machines through a link 206 (via a network) and the HTTP handler 204 couples to remote servers through a link 208 (via a network).

The web server 202 and the HTTP handler 204 also communicate with one another as well as with various supporting modules and a data storage device 210. The data storage device 210 provides persistent or non-volatile storage for various data items being maintained by the intermediary server 200.

- 5 Typically, for each user or requestor associated with a client machine, the data storage device provides separate storage.

The supporting modules include a session manager 212 that manages a communication session between a client machine and the intermediary server 200 as well as between the intermediary server 200 and a remote  
10 server. A cookie manager 214 manages "cookies" such that those being received from a remote server are stored to the data storage device 210 and those "cookies" previously stored in the data storage device 210 are delivered to the remote server at appropriate times. More generally, "cookies" refer to server stored information. Such server stored information is often set by a  
15 remote server and used for session, state or identification purposes. An application plug-in framework 216 provides support for plug-in modules that provide additional functionality to the intermediary server 200. The plug-in modules can perform a variety of different types of functions and thus provide clients or servers with additional functions or features. The plug-in modules  
20 can also be developed by third-parties which can then take advantage of the infrastructure provided by the intermediary server 200 and the information retrieval system 100. The intermediary server 200 also includes an HTML parser 218 which is a processing module that is used to parse requested content received from a remote server and then modify the content (e.g.,  
25 HTML content) in predetermined ways.

FIGs. 2B and 2C illustrate an example of a computer system that may be used in accordance with the invention. The computer system can, for example, correspond to any of the client machines 104 and 106, the intermediary server 108, or the remote servers 110 and 112. FIG. 2B shows  
30 a computer system 221 that includes a display 223, screen 225, cabinet 227, keyboard 229, and mouse 231. Mouse 231 may have one or more buttons

for interacting with a graphical user interface. The cabinet 227 houses a removable medium (e.g., CD-ROM) drive 233, system memory and a hard drive (see FIG. 2C) which may be utilized to store and retrieve software programs incorporating computer code that implements the invention, data for use with the invention, and the like. Although CD-ROM 235 is shown as an exemplary computer readable storage medium, other computer readable storage media including floppy disk, tape, flash memory, system memory, and hard drive may be utilized. Additionally, a data signal embodied in a carrier wave (e.g., in a network including the Internet) may be the computer readable storage medium. In one implementation, an operating system for the computer system 221 is provided in the system memory, the hard drive, the CD-ROM 235 or other computer readable storage medium and serves to incorporate the computer code that implements the invention.

FIG. 2C shows a system block diagram of the computer system 221 used to perform the processing of an embodiment of the invention. As in FIG. 2C, the computer system 221 includes monitor 223 and keyboard 229, and mouse 231. The computer system 221 further includes subsystems such as a central processor 251, system memory 253, fixed storage 255 (e.g., hard drive), removable storage 257 (e.g., CD-ROM disk), display adapter 259, sound card 261, speakers 263, and network interface 265. The central processor 251, for example, can execute computer program code (e.g., an operating system) to implement the invention. An operating system is normally, but necessarily, resident in the system memory 253 during its execution. Other computer systems suitable for use with the invention may include additional or fewer subsystems. For example, another computer system could include more than one processor 251 (i.e., a multi-processor system) or a cache memory.

The system bus architecture of computer system 221 is represented by arrows 267. However, these arrows are illustrative of any interconnection scheme serving to link the subsystems. For example, a local bus could be utilized to connect the central processor to the system memory and display adapter. The computer system 221 shown in FIG. 2B is but an example of a

computer system suitable for use with the invention. Other computer architectures having different configurations of subsystems may also be utilized.

FIG. 3 is a flow diagram of intermediary request processing 300 according to one embodiment of the invention. The intermediary request processing 300 is, for example, performed by the intermediary server 108 illustrated in FIG. 1.

The intermediary request processing 300 begins with a decision 302 that determines whether a resource request has been received from a client machine. When the decision 302 determines that a resource request has not yet been received, then the intermediary request processing 300 awaits such a request. Once the decision 302 determines that a resource request has been received, then an address for the remote server having the requested resource is determined 304. The address is determined 304 from at least the resource request that has been received. Next, "cookies" (server stored information) associated with the remote server are obtained 306 from central storage. Typically, the "cookies" are maintained in the central storage by the intermediary server in a manner that associates the "cookies" with the requestor or user as well as the remote server. Then, a request for the requested resource with the associated "cookies" are sent 308 to the remote server.

At this point, the intermediary request processing 300 awaits a response to the request. In particular, a decision 310 determines whether a response has been received. When the decision 310 determines that a response has not yet been received, the intermediary request processing 300 awaits such a response. Once the decision 310 determines that a response has been received, then any "cookies" provided with the response are saved 312 to the central storage. Again, the "cookies" are preferably stored in the central storage such that they are associated with not only the user or requestor but also the remote server.

Next, the response is modified 314 so that internal links point to the intermediary server instead of one or more remote servers. After the response has been modified 314, the modified response is saved 316 to the central storage. By saving the modified response, a historical database of  
5 accessed resources can be maintained for the user or requestor and thus enabling subsequent retrieval of previously viewed resources. In an alternative embodiment, the unmodified response can be saved 316 to the central storage. The modified response is then sent 318 to the requestor. At this point, the user or requestor has received the response initially requested,  
10 though the response has been modified. By modifying the response, the intermediary server is able to insert itself between the client and remote server so as to provide the user with an enhanced environment for accessing remote resources. After the modified response is sent 318 to the requestor, the intermediary request processing 300 is complete and ends.

FIGs. 4A and 4B are flow diagrams of client processing 400 according  
15 to one embodiment of the invention. The client processing 400 is, for example, performed by a client machine, such as the client machine 104 or the client machine 106 illustrated in FIG. 1. The client processing 400 can be considered as having two portions, a first portion representing an initial client  
20 processing illustrated in FIG. 4A and a second portion representing page request client processing illustrated in FIG. 4B.

The client processing 400 initially begins by a user interacting with a client machine to request 402 a login page. The login page is utilized to  
25 permit a user interacting with the client machine to log onto an intermediary server so that the intermediary server can confirm that the user is an authorized user. After the login page is requested 402, a decision 404 determines whether the login page has been received. Here, the client processing 400 awaits the receipt of the login page from the intermediary  
30 server. When the decision 404 determines that the login page has not yet been received, the client processing 400 awaits receipt of the login page. Once the decision 404 determines that the login page has been received, the login page is displayed 406. Here, the login page can be presented to a user

on a display screen associated with the client machine. Once the login page is displayed 406, a user can enter login information 408. For example, the login page is typically displayed by a browser application that allows a user to enter information into the login page through use of a keyboard and/or pointing device. Next, a login to the intermediary server is requested 410.

A decision 412 then determines whether the client processing 400 has been notified of the successful login by the intermediary server. When the decision 412 determines that the login request has been declined by the intermediary server, then a login failed message is displayed 414. Thereafter, the client processing 400 can return to repeat the operation 406 and subsequent operations so that the login can be again attempted. On the other hand, when the decision 412 determines that the login request has been successful, a session identifier that has been returned from the intermediary server is stored 416. Here, the session identifier is stored (e.g., as a "cookie") on the client machine. Next, an initial webpage from the intermediary server is received and displayed 418.

The characteristics of the initial webpage are that at least some of the hyperlinks embedded therein are initially directed to the intermediary server instead of a remote server that hosts the content (resources) associated with the hyperlink. At this point, a user (or requestor) can select hyperlinks provided on the displayed webpage from the intermediary server. Further discussion below assumes that the hyperlinks of interest are all directed to the intermediary server instead of to remote servers where their content resides.

A decision 420 then determines whether a hyperlink on the displayed webpage has been selected 420. When the decision 420 determines that a hyperlink has not been selected, the client processing 400 awaits such a selection. Once the decision 420 determines that a hyperlink has been selected, the client processing 400 continues. A host name lookup is performed 422 to obtain the Internet Protocol (IP) address associated with the selected hyperlink. Typically, a Domain Name Service (DNS) server



associated with the Internet is used to obtain the IP address. Next, a connection is opened (or maintained if already opened) between the client machine and the intermediary server. The connection can utilize security measures such as Secure Sockets Layer (SSL) if desired. A request for a URL associated with the selected hyperlink together with the session identifier are then sent 426 to the intermediary server. The URL identifies the content (resource) on a remote server being requested by the selected hyperlink.

Next, a decision 428 determines whether a response has been received from the intermediary server. When the decision 428 determines that a response has not yet been received, a decision 430 determines whether a time-out has occurred. When the decision 430 determines that a time-out has occurred, then the client processing 400 returns to repeat the operation 420 and subsequent operations because the previous request for content associated with selected hyperlink was not successful. Alternatively, when the decision 430 determines that a time-out has not yet occurred, then the client processing 400 returns to repeat the decision 428 so as to continue to wait for the receipt of the response from the intermediary server.

On the other hand, when the decision 428 determines that a response has been received, then the response received from the intermediary server is displayed 432. In addition, a decision 434 determines whether there are additional fetches of content to be performed in fully rendering the response. For example, responses (e.g., such as web pages) often include source links for additional content, such as graphics or advertisements, that are obtained by additional fetches (requests). Hence, when the decision 434 determines that there are additional fetches to be made, then an additional fetch request is sent 436 to the intermediary server. Thereafter, the client processing returns to repeat the operation 428 and subsequent operations. On the other hand, when the decision 434 determines that there are no additional fetches to be made, then the client processing 400 returns to repeat the operation 420 and subsequent operations so that additional hyperlink selections can be processed. The client processing 400 ends when the session ends which can

occur through stopping the browser application, shutting down or restart of the client machine, or due to prolonged inactivity.

FIGs. 5A - 5D are flow diagrams of intermediary processing 500 according to one embodiment of the invention. The intermediary processing 500 is, for example, performed by an intermediary server, such as the intermediary server 108 illustrated in FIG. 1.

The intermediary processing 500 begins with a decision 502 that determines whether a connection request has been received. When the decision 502 determines that a connection request has been received, a connection is established 504 (or maintained if previously existing) with the requestor (client machine). The connection can be either a secure connection or an unsecure connection. For example, in the case of a secure connection, in one embodiment, a SSL handshake can be performed to exchange encryption related information. Following operation 504, the intermediary processing 500 returns to repeat the decision 502 so that a subsequent request can be processed. On the other hand, when the decision 502 determines that a connection request has not been received, then a decision 506 determines whether a URL request has been received. When the decision 506 determines that a URL request has not been received, then the intermediary processing 500 returns to repeat the decision 502 and subsequent operations.

Alternatively, when the decision 506 determines that a URL request has been received, then the intermediary processing 500 determines the type of URL being requested. In general, the URL can be associated with a remote server (remote URL) or the intermediary server (local URL). In this embodiment of the intermediary processing 500 it is assumed that a login page and a view history search page are available from the intermediary server and that various other URLs are available from remote servers. However, it should be recognized that the intermediary server can support various additional pages in a similar fashion.

More particularly, when the decision 506 determines that a URL request has been received, then a decision 508 determines whether the URL request is a login page request from a requestor. In this embodiment, the requestor is assumed to be a client machine, though the requestor could also  
5 be a user of the client machine alone or in combination with the client machine. When the decision 508 determines that a login page request has been received, the login page is sent 510 to the requestor. Following the operation 510, the intermediary processing 500 returns to repeat the decision 502 and subsequent operations.

10 Alternatively, when the decision 508 determines that a login page request has not been received, a decision 512 determines whether the session is authorized. When the decision 512 determines that the session is not authorized, then the intermediary processing 500 returns to repeat the operation 510 and subsequent operations so that the requestor can be forced  
15 to login and thus establish an authorized session before any (non-login) URL requests are processed. On the other hand, when the decision 512 determines that the session is authorized, then a decision 514 determines whether the URL request is a view history request. When the decision 514 determines that the URL request is a view history request, then a view history search page is sent 515 to the requestor. Following the operation 515, the  
20 intermediary processing 500 returns to repeat the decision 502 and subsequent operations. Alternatively, when the decision 514 determines that the URL request is not a view history request, then the intermediary processing 500 assumes that the URL requested is a remote URL request for  
25 content (resource) (e.g., web page) at a remote server and thus not a local URL request.

The intermediary processing 500 then continues for the remote URL request with preprocessing of the URL request. Initially, a decision 516  
30 determines whether the URL request is a secure request. When the decision 516 determines that the URL request is not a secure request (i.e., an unsecure request), then the host name associated with the request is modified 518 to remove the intermediary name. Then, a host name lookup is

performed 520 to obtain an IP address of the appropriate remote server. A connection is then opened 522 (or maintained if already opened) to the remote server. Next, "cookies" associated with the modified host name are obtained 524. At this point, the intermediary server has completed  
5 preprocessing of the URL request and is prepared to now forward the request to the remote server.

Alternatively, when the decision 516 determines that the URL request is a secure request, the preprocessing of the URL request by the intermediary server is performed differently. Initially, the host name for the appropriate  
10 remote server is obtained 526. In one embodiment, the host name can be obtained 526 from storage. Here, the storage can, for example, be the data storage device 210 illustrated in FIG. 2. In another embodiment, the host name can be obtained 526 from the URL request. After the host name for the appropriate remote server is obtained 526, a host name lookup is performed  
15 528 to obtain an IP address of the appropriate remote server. A connection is then opened 530 (or maintained if already opened) to the remote server. Next, a secure handshake is performed 532 between the intermediary server and the remote server as needed. The "cookies" associated with the obtained host name are then obtained 534. Following the operation 534, the  
20 preprocessing of the URL request at the intermediary server is completed and the request is now able to be forwarded to the remote server.

At this point, regardless of whether processing a secure or unsecure URL request, the request for the URL with associated "cookies" is sent 536 to the remote server. A decision 538 then determines whether a response has  
25 been received. When the decision 538 determines that a response has not yet been received, the intermediary processing 500 awaits such a response. Once the decision 538 determines that a response has been received, then a decision 540 determines whether "cookies" are present in the response. When the decision 540 determines that "cookies" are present in the response,  
30 then the "cookies" are extracted 542 from the response. The extracted "cookies" are then saved 544. Typically, the "cookies" are stored in central storage provided within or associated to the intermediary server. Following

the operation 544, as well as following the decision 540 when it is determined that "cookies" are not present in the response, then a host name portion of any URLs within headers of the response are modified 546.

A decision 548 then determines whether the response is a HTML response. Here, in general, a response can be of a variety of forms such as HTML, graphics, .pdf, MPEG, or other various formats. When the decision 548 determines that the response is not a HTML response, then the response can be immediately sent (or forwarded) 550 to the requestor. Then, a decision 552 determines whether the response is completed. When the decision 522 determines that the response is completed, then the intermediary processing 500 returns to repeat the decision 502 and subsequent operations so that additional requests can be processed. On the other hand, when the decision 552 determines that so far only a portion of the response has been sent to the requestor, then the intermediary processing 500 returns to repeat the decision 538 and subsequent operations so that subsequent portions of the response can be similarly processed.

On the other hand, when the decision 548 determines that the response is an HTML response, then the response is processed 554 through any application plug-in(s) that are installed or present at the intermediary server. Next, the resulting HTML is saved 556 with reference to the requestor. Here, the resulting HTML is saved so that the requestor can, at a later date, retrieve previously requested HTML and thus review or search through content that has been previously requested (i.e., one's historical trail) by the requestor. Next, a toolbar HTML can be inserted 558 into the resulting HTML. The toolbar that is produced by the toolbar HTML can provide controls or content that are added to the resulting HTML so as to facilitate features or functionality provided by the intermediary server. Next, certain URLs within the resulting HTML can be modified 560. In one embodiment, the modifications 560 to the certain URLs can be achieved by modifying the host name portion of URLs within certain tags of the resulting HTML. In another embodiment, the modifications 560 to the certain URLs can be achieved by adding suffixes to the certain URLs. The suffixes thus serve to

allow the URLs to carry additional information. Further, certain URLs provided or produced by scripting language portions within the resulting HTML can be modified 562. Examples of scripting languages include JavaScript and VBscript. In one embodiment, a host name portion of the certain URLs provided or produced by scripting language portions within the resulting HTML are modified 562. In another embodiment, the certain URLs provided or produced by scripting language portions are modified 562 to include suffixes which carry supplemental information. Additional details on modifying scripting language portions is provided below with reference to FIGs. 12A and 12B. Thereafter, the resulting HTML is sent 564 to the requestor.

A decision 566 then determines whether the request has been completed. When the decision 566 determines that the request has been completed, then the intermediary processing 500 returns to repeat the decision 502 and subsequent operations so that additional requests can be processed. On the other hand, when the decision 566 determines that the request is not yet completed, then the intermediary processing 500 returns to repeat the decision 538 and subsequent operations so that remaining portions of the response can be similarly processed upon being received. The intermediary processing 500 can thus operate to process a single response to a URL request in multiple pieces or blocks of data. In such case, the intermediary processing 500 can process a response from a remote server as it arrives so that responsiveness to the requestor is not hindered. In this regard, the intermediary processing 500 causes the operations 538 – 566 to be repeated for each piece or block of data associated with a response.

FIG. 6 illustrates a diagram of an information retrieval system 600 according to one embodiment of the invention. The information retrieval system 600 is generally similar to the information retrieval system 100 of FIG. 1. The operation of the information retrieval system 600 is discussed below with reference to three representative examples which illustrate its operation according to one embodiment. One example pertains to an unsecure request and the other examples pertain to a secure request. The information

retrieval system 600 includes a client 602, an intermediary server 604 with a data store 606, and a remote server 608. It is assumed that the initial client processing portion (e.g., FIG. 4A) has already been performed.

The first representative example pertains to an unsecure request which can be initiated by the user selecting (420) a hyperlink in a displayed webpage. The selected hyperlink is assumed to be

<http://www.xyz.com.danastreet.com/quote/msft>

where "http" is the protocol, "www.xyz.com.danastreet.com" is the hostname with "danastreet.com" being a domain and "xyz.com" being a subdomain, and "/quote/msft" being a path to the particular resource being requested by selection of the hyperlink. Hence, the domain name lookup (422) of the hostname "www.xyz.com.danastreet.com" is resolved to the IP address of danastreet.com. This can be achieved by configuring network routers to give the same IP address for any subdomain of datastreet.com. The request is then sent (426) from the client 602 to the intermediary server 604. The request is, for example, as follows:

```
GET: /quote/msft HTTP/1.0
Host: www.xyz.com.danastreet.com
Cookie: DSID = 123xyzzbc
```

Other information can also be included within the request such as other cookies, encoding-accepted, etc. The cookie is, in this example, a session cookie and is used in determining whether the client 602 is authorized (514) for user with the intermediary server 604.

In the case of an unsecure request, the hostname within the request is modified (518) to remove the "danastreet.com" portion and then perform a domain name lookup (520) on the modified portion (i.e., remaining portion) of the hostname ("www.xyz.com"). Any cookies in the data store 606 previously received on behalf of the user that are associated with the modified hostname are then obtained (524). Next, a request by the intermediary server 604 is sent (536) to the remote server 608. The request is, for example, as follows:

```
GET: /quote/msft HTTP/1.0
```

Host: www.xyz.com  
Cookie: xyzUserID = sam

Other information can also be included within the request. Note that the  
5 cookie provided with the original request pertains to the intermediary server  
604 and thus is not forwarded with the request to the remote server 608.

The remote server 608 receives the request and returns (538) a  
response header and some or all of the content of the requested resource.  
An exemplary response can have the following format:

```
10      200 OK
      Set-cookie: xyzuserID = Samuel, expires = 01-Jul-2002
      Content-type: text/html
      Content-length: 2000
      Location: http://www.xyz.com/stockquote/msft
15      <HTML>

          * * *

20      </HTML>
```

Since the response included a "cookie" to be set, the set-cookie command is  
remove (542) from the response and then saved (544) in the data store 606.  
Next, to the extent they are present, the hostnames within the headers are  
modified (546) to point to the intermediary server 604. In this example, the  
25 location header includes a full path (including hostname), namely,  
http://www.xyz.com/stockquote/msft, which is thus modified to  
http://www.xyz.com.danastreet.com/stockquote/msft. For paths in the  
headers that do not include hostnames, no modifications are needed as the  
browser application operating on the client 602 will cause the current  
30 hostname ("www.xyz.com.danastreet.com") to be used for such paths. Next,  
processing through one or more plug-in(s) can be performed (554) to alter,  
modify or supplement the HTML data accompanying the response. The  
resulting HTML is saved (556) in the data store 606. If desired, a toolbar can  
be inserted (558) into the resulting HTML to facilitate operations or functions  
35 supported by the intermediary server 604. Still further certain URLs within the  
resulting HTML or those produced by scripting languages are modified (560),



562) to point to the intermediary server 604. For example, if the resulting HTML included the following hyperlink:

`<a ref = http://www.xyz.com/quote/msft>`

then the hyperlink would be modified to the following:

5                   `<a ref = http://www.xyz.com.danastreet.com/quote/msft>`

Additional details are provided below on the modification (560, 562) of the certain URLs within the resulting HTML or those produced by scripting languages. Thereafter, the response including the modified HTML can be delivered (564) from the intermediary server 604 to the client 602.

10           The second representative example pertains to a secure request which can be initiated by the user selecting (420) a hyperlink in a displayed webpage. The selected hyperlink is assumed to be

`https://secure.danastreet.com:700/quote/msft`

where "https" is the protocol which uses Secure Sockets Layer (SSL),

15           "www.secure.danastreet.com" is the hostname with "danastreet.com" being a domain and "secure" being a subdomain, "700" indicating a port number, and "/quote/msft" being a path to the particular resource being requested by selection of the hyperlink. Hence, the domain name lookup (422) of the hostname "secure.danastreet.com" is resolved to the IP address of  
20           danastreet.com. For secure communications, the browser in making the connection to the hostname will require an authentication certificate be present for the hostname. The presence of such an authentication certificate for the matching hostname is helpful because a browser attempting such a  
25           secure connection does not alert the user of its absence (e.g., with a pop-up dialog box). Here, single certificate for "secure.danastreet.com" can be used for all secure requests. The request is then sent (426) from the client 602 to the intermediary server 604. The request is, for example, as follows:

30                   GET: /quote/msft HTTP/1.0  
                      Host: secure.danastreet.com:700  
                      Cookie: DSID = 123xyzzbc

Other information can also be included within the request such as other cookies, encoding-accepted, etc. The cookie is, in this example, a session cookie and is used in determining whether the client 602 is authorized (514) for user with the intermediary server 604.

5 In the case of a secure request, the hostname within the request is not able to directly identify the remote server 608 where the request is eventually to be delivered. However, the hostname for the remote server 608 can be obtained (526) from the data storage 606. More particularly, in accordance with one embodiment, the port number and a session (or user) identifier are  
10 used to lookup in the data storage 606 the appropriate hostname. Here, the port number "700" for the client 602 is assumed to represent "xyz.com". Once the appropriate hostname has been obtained (526), a domain name lookup (528) is performed on the hostname ("www.xyz.com"). Next, a connection between the intermediary server 604 and the remote server 608 is opened  
15 (530) or maintained is already opened, and secure handshaking performed (532). Any cookies in the data store 606 previously received on behalf of the user that are associated with the hostname are then obtained (534). Next, a request by the intermediary server 604 is sent (536) to the remote server 608. The request is, for example, as follows:

20 GET: /quote/msft HTTP/1.0  
Host: www.xyz.com  
Cookie: xyzUserID = sam

Other information can also be included within the request. Note that the  
25 cookie provided with the original request pertained to the intermediary server 604 and thus is not forwarded with the request to the remote server 608.

The remote server 608 receives the request and returns (538) a response header and some or all of the content of the requested resource.

An exemplary response can have the following format:

30 HTTP/1.0 200 OK  
Set-cookie: xyzuserID = Samuel, expires = 01-Jul-2002  
Content-type: text/html  
Content-length: 2000  
Location: https://www.xyz.com/stockquote/msft

<HTML>

\* \* \*

</HTML>

Since the response included a "cookie" to be set, the set-cookie command is remove from the response (542) and then saved (544) in the data store 606.

Next, to the extent they are present, the hostnames within the headers are

modified (546) to point to the intermediary server 604. In this example, the location header includes a full path (including hostname), namely, <https://www.xyz.com/stockquote/msft>, which is thus modified to <https://secure.danastreet.com:700/stockquote/msft>. For paths in the headers

that do not include hostnames, no modifications are needed as the browser application operating on the client 602 will cause the current hostname with port number ("[www.secure.danastreet.com:700/](https://www.secure.danastreet.com:700/)") to be used for such paths. Next, processing through one or more plug-in(s) can be performed (554) to alter, modify or supplement the html data accompanying the response. The resulting html is saved (556) in the data store 606. If desired, a toolbar can be inserted (558) into the resulting html to facilitate operations or functions supported by the intermediary server 604. Still further the hostname portion of URLs within certain tags within the resulting html or those produced by scripting languages are modified (560, 562) to point to the intermediary server 604. For example, if the resulting html included the following hyperlink:

<a ref = <https://www.ijk.com/quote/msft>>

then the hyperlink would be modified to the following:

<a ref = <https://secure.danastreet.com:701/quote/msft>>

As another example, the resulting html could also include hyperlinks not requiring a secure connection and thus be converted as noted above in the first representative example. Additional details are provided below on the modification (560, 562) of the certain URLs within the resulting HTML or those produced by scripting languages. Thereafter, the response including the

modified HTML can be delivered (564) from the intermediary server 604 to the client 602.

The third representative example pertains to a secure request which can be initiated by the user selecting (420) a hyperlink in a displayed webpage. The selected hyperlink is assumed to be

<https://secure.danastreet.com/quote/msft:danainfo:host=www.xyz.com>

where "https" is the protocol which uses Secure Sockets Layer (SSL), "secure.danastreet.com" is the hostname with "danastreet.com" being a domain and "secure" being a subdomain, "/quote/msft" being a path to the particular resource being requested by selection of the hyperlink, "danainfo" is a keyword, and "www.xyz.com" is the host where the requested resource resides. Hence, the domain name lookup (422) of the hostname "secure.danastreet.com" is resolved to the IP address of danastreet.com. The request is then sent (426) from the client 602 to the intermediary server 604. The request is, for example, as follows:

```
GET: /quote/msft:danainfo:host=www.xyz.com HTTP/1.0
Host: secure.danastreet.com
Cookie: DSID = 123xyzabc
```

Other information can also be included within the request such as other cookies, encoding-accepted, etc. The cookie is, in this example, a session cookie and is used in determining whether the client 602 is authorized (514) for user with the intermediary server 604.

In the case of a secure request, the hostname within the request is not able to directly identify the remote server 608 where the request is eventually to be delivered. However, the hostname for the remote server 608 is obtained (526) from information provided with the request. More particularly, the information (i.e., host variable) provided as a suffix with the request. In this example, the suffix includes the information that the hostname of the remote server 608 is "www.xyz.com". Once the appropriate hostname has been obtained (526), a domain name lookup (528) on the hostname ("www.xyz.com") is performed. Next, a connection from the intermediary

server 604 and the remote server 608 is opened (530) or maintained if already opened, and secure handshaking is performed (532). Any cookies in the data store 606 associated with the hostname and the requestor are then obtained (534). Next, a request by the intermediary server 604 is sent (536) to the remote server 608. The request is, for example, as follows:

```
GET: /quote/msft HTTP/1.0
Host: www.xyz.com
Cookie: xyzUserID = sam
```

- 10 Other information can also be included within the request. Note that the cookie provided with the original request pertained to the intermediary server 604 and thus is not forwarded with the request to the remote server 608.

The remote server 608 receives the request and returns (538) a response header and some or all of the content of the requested resource.

- 15 An exemplary response can have the following format:

```
HTTP/1.0 200 OK
Set-cookie: xyzuserID = Samuel, expires = 01-Jul-2002
Content-type: text/html
Content-length: 2000
Location: https://www.xyz.com/stockquote/msft
<HTML>
```

\* \* \*

- 25 </HTML>

- Since the response included a "cookie" to be set, the set-cookie command is remove from the response (542) and then saved (544) in the data store 606. Next, to the extent they are present, the hostnames within the headers are modified (546) to point to the intermediary server 604. In this example, the location header includes a full path (including hostname), namely, https://www.xyz.com/stockquote/msft, which is thus modified to https://secure.danastreet.com/stockquote/msft:danainfo:host=www.xyz.com. In this example, not only are the hostnames modified but also variables are added to the end (i.e., suffix) of the URL. With this example, the relative URLs need to be modified to include the variable information

("danainfo:host=www.xyz.com") at the end of the relative URLs. The hostnames for the relative URLs are properly provided by the browser application operating on the client 602 will cause the current hostname ("secure.danastreet.com") to be used for such paths. Next, processing through one or more plug-in(s) can be performed (554) to alter, modify or supplement the html data accompanying the response. The resulting HTML is saved (556) in the data store 606. If desired, a toolbar can be inserted (558) into the resulting HTML to facilitate operations or functions supported by the intermediary server 604. Still further the hostname portion of URLs within certain tags within the resulting HTML or those produced by scripting languages are modified (560, 562) to point to the intermediary server 604. For example, if the resulting HTML included the following hyperlink:

```
<a ref = https://www.xyz.com/quote/msft>
```

then the hyperlink would be modified to the following:

```
<a ref=https://secure.danastreet.com/quote/msft:danainfo:host=www.xyz.com>
```

Also, if the resulting html include the following relative hyperlink:

```
<a ref = a.html>
```

then the hyperlink would be modified to the following:

```
<a ref = a.html:danainfo:host=www.xyz.com>
```

As another example, the resulting HTML could also include hyperlinks not requiring a secure connection and thus be converted as noted above in the first example. Thereafter, the response including the modified HTML can be delivered (564) from the intermediary server 604 to the client 602. In any case, one advantage of this third exemplary approach is that the browser application at the client machine can execute scripting language instructions to perform the modifications, and thus alleviates some processing burden from the intermediary server.

The application plug-ins can operate to perform a variety of functions. Typically, the various application plug-ins can be considered as filters that operate in a serial fashion stacked one upon another and serve to alter or

modify the HTML data. The modifications (or filtering) can serve to translate the HTML data between languages, remove advertisements from the HTML pages, provide customization for the requestors, etc.

FIG. 7 is a diagram of application plug-in processing 700 according to one embodiment of the invention. The application plug-in processing 700 is, for example, associated with application plug-ins that have been registered with the application plug-in framework 216 illustrated in FIG. 2A. Each application plug-in serves to, in some manner, alter, modify or supplement the HTML data associated with the response. The application plug-in processing 700 is, for example, performed by the operation 554 illustrated in FIG. 5D. The application plug-in processing 700 typically receives the HTML data associated with the response (though it may have already been partially modified, e.g., by operation 546 illustrated in FIG. 5C). In any case, the HTML data received is sent through one or more plug-in modules. Each of the plug-in modules can be provided by a different third-party to alter, modify or supplement the HTML data in different ways. In the embodiment shown in FIG. 7, the incoming HTML data is first sent to application plug-in #1 702, and then the resulting HTML data is passed through application plug-in #2 704, and then the resulting HTML data is supplied to an application plug-in #3 706. The modified HTML data output from the application plug-in #3 706 represents the HTML data after passing through the series of application plug-ins. In this manner, multiple third-party applications can be provided and supported by the intermediary server 100, 200. This allows different applications to alter the HTML data in different ways so that multiple features or services can be supported by the platform or architecture provided by the information retrieval system 100, namely, the intermediary server 108, 200.

In accordance with another aspect of the invention, the intermediary server (e.g., intermediary server 108, 200) operates to store remote resources that have been requested by various requestors. As an example, operation 556 of FIG. 5D pertains to saving resulting HTML with reference to the particular requestor. In general, the resources (e.g., HTML) can be stored

in a data storage device such as the data storage device 210 illustrated in FIG. 2A. More particularly, however, the data storage device 210, can include a database and a file server. FIG. 8 illustrates a database 800 and a file server 802 according to one embodiment of the invention. The database 800 and the file server 802 together operate to provide intelligent data storage for resources that have been requested by requestors through the intermediary server. The database 800 illustrated in FIG. 8 stores the URL's that have been previously requested by various users. For example, as shown in FIG. 8, for user-A, the database 800 stores the URL, host name, path, timestamp, and a file reference for each remote resource request. The timestamp indicates the time at which the URL was requested or received. The intermediary server can set the timestamp or the requestor can set the timestamp. The file reference is a unique reference to a file residing within the file server 802. The file stored within the file server 802 is the response (e.g., resulting HTML or content) that was previously viewed by the requestor.

By providing the storage of resources that requestors have previously obtained through the intermediary server, the intermediary server can thus serve as storage of a history of resources that various users (requestors) have previously retrieved. This allows requestors to subsequently request from the intermediary server those resources that the users have previously requested through the intermediate server.

The retrieval of such previously requested resources can be facilitated by a view history search page (e.g., operation 515 of FIG. 5A). The view history search page can enable a requestor to search through the resources (pages, documents, etc.) that the requestor has previously obtained through the intermediary server to thereby identify one or more particular previously viewed resources. The searching can include text searching of the resources themselves. The searching can also be facilitated by filtering on any of the parameters stored within the database, such as hostname or timestamp. In addition, the database could also store keywords or phrases for each or some of the resources, to thereby facilitate searching through such keywords or



phrases. The timestamp can be used so that the storage of identical URL's (resources) can be achieved but distinguished the resources by their timestamp. This allows the same resource to be saved multiple times (even for a particular requestor) and be distinguished by the different timestamps.

- 5 The content of the resource saved at the different times may or may not be different. Hence, the view history search page could also link the URL's that are the same such that requestors are given the opportunity to retrieve the other versions of the same resource that have been stored at different times.

- Still further, a resource request typically has internal links that retrieve  
10 other resources that form a portion of the overall resource. For example, a webpage can contain a link (e.g., internal link or hyperlink) to another resource location to retrieve an image. This is commonly done for providing a banner advertisement for the webpage. These internal links are also stored within the database in accordance with one embodiment of the invention.
- 15 Hence, if a requestor desires to retrieve a resource that has been previously stored, the webpage is typically initially requested and then retrieved from the data storage device. Once the webpage is received, the requestor (e.g., network browser operating on the client machine) operates to request and then retrieve the internal links from the data storage device. If, for some  
20 reason, the content associated with the internal links is not available from the data storage device, then the user can be so informed and offered the opportunity to retrieve the content from the remote server or other options.

- Another aspect of the invention concerns the insertion of a toolbar into the response (i.e., HTML data) provided by the remote server before the  
25 response is delivered to the requestor. Such toolbar insertion can, for example, be performed at operation 558 of FIG. 5D. Alternatively, such toolbar insertion can be performed by an application plug-in such as shown in FIG. 7.

- The toolbar that is inserted within the webpage (e.g., HTML data) can  
30 take a variety of different forms. More particularly, the toolbar is inserted into

the HTML data of the webpage by inserting toolbar HTML into the HTML data. In one embodiment, the toolbar HTML includes graphical and/or textural content for the toolbar that is displayed when the HTML data is displayed when rendered on the client machine via a network browser. The toolbar HTML also includes one or more links to other resources. The links can be graphical or textual and thus be associated with the graphical and/or textural content. In one embodiment, the toolbar HTML also includes an activation script. The activation script is a script programming language understood by the network browser on the client machine. The activation script is executed by the browser at the client machine when seeking to render the toolbar HTML for the requestor. Examples of the script programming language include JavaScript and VBscript. The toolbar HTML can be inserted into the HTML data in a variety of different positions. In one embodiment, the toolbar HTML is inserted into the HTML data directly following the initial body tag (<body>) so as to place the toolbar at the top portion of the rendered HTML page.

The insertion of the toolbar into web pages can be relatively straightforward or complex, depending upon the particular webpage. However, for a general solution, the toolbar insertion operations need to be intelligent enough to determine when it is appropriate to insert the toolbar and when not too. In this regard, the webpage (e.g., HTML data) is modified to insert JavaScript code following a body tag within each frame associated with the webpage. When executed at the client side, the JavaScript code determines if it is currently within a frame. If it is not currently in a frame, then the toolbar can simply be inserted. On the other hand, when the JavaScript code determines that it is within a frame, then it examines the size of the frame to determine whether the frame is large enough to support the toolbar being inserted. In other words, the JavaScript code for insertion of the toolbar is inserted into every page, indeed every frame of every page, but is only displayed in certain pages or frames.

FIG. 9A illustrates a toolbar activation process 900 according to one embodiment of the invention. The toolbar activation process 900 is processing associated with an activation script. The toolbar activation process 900 is performed by the network browser at the client machine when the resulting HTML is being rendered by the network browser. The toolbar activation process 900 begins with a decision 902 that determines whether the activation script being executed is within a frame of the resulting HTML. The resulting HTML pertains to a webpage that is displayed by the network browser. Web pages, or their HTML description, can define none, one or multiple frames. The toolbar HTML is inserted into the webpage (resulting HTML) at least once and, if one or more frames are defined, inserted into each of the frames. When the decision 902 determines that the activation script being executed is not within a frame, then the toolbar can be rendered 904 as part of the webpage produced by the network browser. On the other hand, when the decision 902 determines that the activation script is within a frame, then a decision 906 determines whether the size of the frame is greater than a threshold size. Here, the size of the current frame is examined to determine whether it is of sufficient size (e.g., height and width) to properly support the toolbar. When the decision 906 determines that the size of the frame is greater than the threshold size, then the toolbar activation process 900 has determined that the frame is sufficiently large enough to support the toolbar and thus renders 904 the toolbar via the network browser. On the other hand, when the decision 906 determines that the size of the frame is not greater than the threshold size, then the toolbar is not rendered and the toolbar activation process 900 is complete and ends. Accordingly, the toolbar activation process 900 can be performed at least once for the webpage and multiple times if frames are present, such that each instance of the toolbar activation process 900 dynamically self-determines whether the toolbar instance should be rendered within the particular webpage or frame thereof.

An example of an activation script for performing operations 902 and 906 of the toolbar activation process 900 is as follows:

```

If (self==top ||
(document.body?(document.body.clientWidth>400
&&document.body.clientHeight>200) :
(self.innerWidth>400&&self.innerHeight>200))) [toolbar
rendering HTML]

```

Note that the toolbar rendering HTML is then written to the network browser to render the toolbar when the "if" statement conditions are met. Here, there are two alternative statement conditions, one for Netscape Navigator network browsers and another for Microsoft Internet Explorer network browsers.

FIG. 9B is a representative toolbar 920 that can be inserted into a webpage in accordance with one aspect of the invention. The representative toolbar 920 is, for example, inserted into the top or bottom of web pages. As noted above, in one embodiment of the invention, the representative toolbar 920 can be rendered in each of the frames of multi-frame web pages that are sufficiently large enough to support the representative toolbar 920.

The representative toolbar 920 includes a series of graphical links 922-936 that are displayed by a network browser. Upon selection of one of the graphical links 922-936, the network browser can be directed to display another resource. The graphical links 922-936 are hyperlinks to predetermined resources. The graphical link 922 represents a company logo "Dana Street" for the company that provides the intermediary server processing. Upon a user clicking on the graphical link 922, the customized homepage for the user provided by the intermediary server can be displayed. The graphical link 924 can display the name of the logged-on user (e.g., "sam") and can also be linked to the user's homepage within the system. The graphical link 926 represents a history management application. Upon selecting the graphical link 926, a history search page can be displayed for the user. In addition, the graphical link 926 can also indicate whether the history management application for the user is "on" or "off" as well as permit the user to toggle between "on" and "off". A graphical link 928 represents a graphical icon associated with a third party application that provides

information about the current site being viewed (e.g., yahoo). Here, the third party application pertains to an external service in which the "A" represents the logo of the external service provider (e.g., <http://www.alexa.com>). The graphical link 930 is an icon that links to a third party application that provides comparative purchasing options. Here, the external service providing the comparison information is, for example, <http://www.clickthebutton.com>. The graphical link 932 is a link to a translation application. As shown in FIG. 9B, the graphical link 932 also indicates that the translator is active in translating to a particular target language (e.g., German in this example). Selecting the graphical link 932 causes a dialog to appear such that a user can turn on and off the translation or switch target languages. The graphical link 934 is a link to a third party application that manages bookmarks on the Internet. The representative external service in this example is Yahoo! Bookmarks (<http://bookmarks.yahoo.com>). Hence, this graphical link 934, in particular, illustrates that third party applications, such as yahoo applications, which appear on a yahoo page toolbar can also be integrated into the representative toolbar 920 without change to their backend service. The graphical link 936 enables a user to log-out from the intermediary server.

The representative toolbar 920 illustrated in FIG. 9B is only one example of a toolbar. It should be recognized that many different toolbars can be used with the toolbar activation process 900. The toolbars can have various different appearances and can provide access to various different services, applications or features.

FIG. 10 is a flow diagram of URL modification processing 1000 according to one embodiment of the invention. The URL modification processing 1000 is, for example, processing performed by operation 560 of FIG. 5D. The URL modification processing 1000 can, for example, be performed by the HTML parser 218 and/or the HTTP handler 204 illustrated in FIG. 2A.

The URL modification processing 1000 begins by selecting 1002 a target URL within the resulting HTML (webpage). Typically, one or more

target URLs are previously identified by scanning the resulting HTML data. Then, a decision 1004 determines whether the target URL is a relative URL. When the decision 1004 determines that the target URL is not a relative URL, then a decision 1006 determines whether the target URL is associated with a secure request (e.g., HTTPS). When the decision 1006 determines that the target URL is not associated with a secure request, then a predetermined hostname is appended 1008 to the hostname provided in the target URL. In other words, the hostname originally provided for the target URL is effectively rewritten such that the original hostname becomes a sub domain and the predetermined hostname becomes the domain for the target URL. Next, a decision 1010 determines whether a port number is specified in the target URL. When the decision 1010 determines that a port number is not specified in the target URL, no port number processing is needed so a decision 1012 then determines whether more target URLs to be processed. As previously noted, these target URLs can have been previously identified by scanning the resulting HTML data. When the decision 1012 determines that there are more target URLs, then the URL modification processing 1000 returns to repeat the operation 1002 and subsequent operations so that additional target URLs can be processed. Alternatively, when the decision 1012 determines that there are no more target URLs, then the URL modification processing 1000 is complete and ends.

On the other hand, when the decision 1006 determines that the target URL is associated with a secure request, then a hostname suffix is added 1014 to the target URL. Then, the hostname of the target URL is replaced 1016 with a predetermined hostname. Following operation 1016, the URL modification processing 1000 proceeds to the decision 1010.

When the decision 1010 determines that a port number is specified in the target URL, then a port number suffix is added 1018 to the target URL. Next, the port number is removed 1020 from after the hostname in the target URL. Following the operation 1020, the URL modification processing 1000 performs the decision 1012.

On the other hand, when the decision 1004 determines that the target URL is a relative URL, a decision 1022 determines whether the source URL has a hostname or port suffix. The source URL is the URL associated with the webpage (including the resulting HTML) that includes the target URL.

- 5 When the decision 1022 determines that the source URL does have either a hostname or a port suffix, then the hostname and/or the port suffix are appended 1024 to the target URL. Following the operation 1024, as well as following the decision 1022 when the source URL does not have a hostname or a port suffix, the URL modification processing 1000 performs the decision
- 10 1012.

Additionally, although not shown in FIG. 10, it should be noted that the URL modification processing can also operate to add a unique identifier to the URL as a port number as discussed above.

- FIG. 11 is a flow diagram of a script modification processing 1100 according to one embodiment of the invention. The script modification processing 1100 is, for example, performed by operation 562 illustrated in FIG. 5D. In general, the script modification processing 1100 operates to modify script portions within the resulting HTML.
- 15

- The script modification processing 1100 initially scans 1102 the HTML data (e.g., of the resulting HTML) for a <script> tag. A decision 1104 then determines whether a script has been found. When the decision 1104 determines that a script has not been found, then a decision 1106 determines whether there is more HTML data to be scanned. When the decision 1106 determines that there is more HTML data to be scanned, then the script modification processing 1100 returns to repeat the operation 1102 and subsequent operations. Alternatively, when the decision 1106 determines that there is no more HTML data to be scanned, the script modification processing 1100 is complete and ends.
- 20
- 25

- On the other hand, when the decision 1104 determines that a script has been found, then the script is searched 1108 to locate text strings "http://"
- 30

or "https://" followed by a hostname. A decision 1110 then determines whether a URL hostname has been found by the searching 1108 of the script. When the decision 1110 determines that a URL hostname has not been found, then a decision 1112 determines whether the end of the script has been reached. When the decision 1112 determines that the end of the script has not yet been reached, then the script modification processing 1100 returns to repeat the operation 1108 and subsequent operations. Alternatively, when the decision 1112 determines that the end of the script has been reached, then the script modification processing 1100 returns to repeat the operation 1102 and subsequent operations so that additional scripts can be found and processed.

On the other hand, when the decision 1110 determines that a URL hostname has been found, then a rewritten hostname is produced 1114. For example, if the URL hostname that was found is "xyz.com", the rewritten hostname that is produced 1114 can, for example, be "xyz.com.danastreet.com". The hostname provided within the script is then replaced 1116 with the rewritten hostname. Following the operation 1116, the script modification processing 1100 returns to repeat the operation 1108 and subsequent operations so that additional hostnames within the script can be similarly processed.

FIGS. 12A and 12B are flow diagrams of a script modification processing 1200 according to another embodiment of the invention. The script modification processing 1200 is, for example, performed by operation 562 illustrated in FIG. 5D. In general, the script modification processing 1200 operates to modify script portions within the resulting HTML.

The script modification processing 1200 initially scans 1201 the HTML data (e.g., of the resulting HTML) for a <script> tag. A decision 1202 then determines whether a script has been found. When the decision 1202 determines that a script has been found, then the script is parsed 1204 to determine or locate predetermined properties and functions associated with the script. As decision 1206 then determines whether at least one property or



function has been found in the script. When the decision 1206 determines that at least one property or function has been found, then the script modification processing 1200 continues such that the script is modified with respect to the properties or functions found within the script so that the script operates as intended even though the intermediary server is interposed between client devices and remote servers.

In particular, for each property or function found within the script, the processing is as follows. A decision 1208 determines whether a selected property or function found within the script pertains to a read of a cookie property. When the decision 1208 determines that the identified property or function does pertain to a read of a cookie property, then the read of the cookie property is replaced 1210 with a `get_cookies` function call. Alternatively, when the decision 1208 determines that the identified property or function is not a read of a cookie property, as well as following the operation 1210, a decision 1212 determines whether the identified property or function pertains to a write to a cookie property. When the decision 1212 determines that the identified property or function does pertain to a write to a cookie property, the write to the cookie property is replaced 1214 with a `set_cookies` functions call.

On the other hand, when the decision 1212 determines that the identified property or function is not associated with a write to a cookie property, as well as following the operation 1214, a decision 1216 determines whether the identified property or function pertains to a write to a property that initiates a request. When the decision 1216 determines that the identified property or function does pertain to a write to a property that initiates a request, then the write to the property that initiates (causes) a request to be replaced 1218 with a `set_URL` function call. Alternatively, when the decision 1216 determines that the identified property or function does not pertain to a write to a property that initiates a request, as well as following the operation 1218, a decision 1220 determines whether the identified property or function pertains to a read from a property that returns a URL. When the decision 1220 determines that the identified property or function does pertain to a read

from a property that returns a URL, then the read from a property that returns a URL is replaced 1222 with an appropriate string.

Furthermore, following the decision 1220 when the identified property or function does not pertain to a read from a property that returns a URL, as well as following the operation 1222, a decision 1224 determines whether more properties or functions were found in the script that still need to be processed. When additional properties or functions have been found and need processing, the script modification processing 1200 returns to repeat the decision 1208 and subsequent operations so that the additional properties or functions can be similarly processed. On the other hand, when the decision 1224 determines that all the properties or functions that have been found within the script have been processed, then the script modification processing 1200 performs a decision 1226. The decision 1226 is also performed when the decision 1202 determines that a script has not been found. The decision 1226 determines whether there is more HTML data to be scanned. When the decision 1226 determines that there is more HTML data to be scanned, then the script modification processing 1200 returns to repeat the operation 1201 and subsequent operations. Alternatively, when the decision 1226 determines that there is no more HTML data to be scanned, the script modification processing 1200 is complete and ends.

Representative examples of a `get_cookies` function, a `set_cookies` function, a `set_URL` function, and string substitution are provided below. These examples are provided to assist understanding and thus should not be deemed restrictions on any aspect of the invention. The following examples use JavaScript as the scripting language.

A first example with respect to the `get_cookies` function and operation 1210 is as follows. In this example, the script includes a script instruction

```
var c = document.cookie;
```

which assigns the cookies associated with the document (page) to the variable `c`. This script instruction would be replaced with

```
var c = get_cookies ("othercookie=abc");
```

which assigns the cookies present on the intermediary server for the particular domain of the document (page) and the particular user (e.g., "othercookie=abc"). In addition, the `get_cookies` function takes the cookies from the intermediary server as its argument and adds to it other cookies that are set by the script.

A second example with respect to the `set_cookies` function and operation 1214 is as follows. In this example, the script includes a script instruction

```
10 document.cookie = "selection=ijk; expires= ...";
```

which stores the cookies associated with the document (page) in the browser. This script instruction (statement) is replaced with

```
document.cookie = set_cookie("<domain>", "selection=ijk; expires= ...");
```

which stores the cookies associated with the document (page) in the browser and also to the intermediary server. The `set_cookie` function includes two arguments. The first argument identifies the domain of the page having the script. The second argument is the value that was originally being assigned to the `document.cookie` property. The `set_cookie` function combines these two arguments and sets a cookie called `servercookieX` with no expiration, where `X` represents a distinguishing numeric value. The browser will cause this cookie to be sent to the intermediary server. The intermediary server can then incorporate the cookie into the cookie storage for the user. The cookie can also be used to expire an existing cookie in storage for the user. Once the cookie is stored at the intermediary server, the next page that the intermediary server returns will cause the `servercookieX` to expire because its no longer needed. Any calls to the `set_cookie` function will also append any cookie values provided within the `servercookieX`.

To further illustrate, consider the following example, where a page from `www.xyz.com` has the following script:

```
30 document.cookie = "a=b";
```

```
var x = document.cookie;
```

Assume also the www.xyz.com server has previously returned a cookie to the intermediary server that has name "id1" with value "sam". The code above will be transformed into:

```
5      document.cookie = set_cookie ("www.xyz.com", "a=b");  
  
      var x = get_cookie ("id1=sam");
```

The first line will cause a cookie "servercookie0" to be set that has the value "a=b ~ domain=www.xyz.com", hence the whole cookie will be:

```
servercookie0 = a=b ~ domain=www.xyz.com
```

10 Note that the domain part of the servercookie0 is used purely by the intermediary server so that it knows which domain is setting the cookie. The second line calls the get\_cookies function which takes its first argument (filled in by the intermediary server while the script was rewritten) and examines all servercookie0's cookies on the browser. It concatenates the first argument  
15 together with any servercookieX cookies, and returns the result:

```
id1=sam; a=b
```

Note, this is the same result that would have been returned from the original page had it not been rewritten.

A third example with respect to the set\_URL function and operation  
20 1218 is as follows. The set\_URL function operates to modify properties that cause a request. In this example, the script includes a script instruction

```
document.location = "http://www.xyz.com/foo.html";
```

which directs the browser to a new page. Such a script instruction can be replaced with

```
25      document.location = set_URL( "", "http://www.xyz.com/foo.html");
```

The set\_URL function call takes two arguments. The first argument is filled in by the intermediary server while the script is being rewritten, and contains any parameters that would normally be provided in a suffix (e.g., "danainfo:") to follow a URL. It is not always needed, as will be explained below. The

second argument is the URL, though it could actually be a script expression (e.g., function call) that assembles or returns a URL.

The set\_URL function examines the URL being set and rewrites it to be of a form that will direct the browser to the intermediary server. As noted above, the modification to URLs can be achieved with various techniques.

If the page is using the hostname modification technique, then relative URLs do not need to be modified since the hostname encodes the necessary information. If the URL is a full URL, then the set\_URL function has all of the information it needs to convert the URL so that either (i) "danastreet.com" is included in the hostname of the URL (hostname modification technique such as for the HTTP case), or (ii) a suffix (e.g., ":danalInfo:host=xxx") is appended to the URL (the HTTPS case). Thus, if the page that the script appears on is using the hostname modification technique, the first argument is not needed by the set\_URL function.

Alternatively, if the page upon which the script is present is using the URL suffix technique, then a relative URL that is passed to the set\_URL function needs to have the same suffix applied to it. In this case, the intermediary server will insert as the first argument to the set\_URL function any arguments that need to be passed in the suffix. For example, if the URL of the page is:

`https://secure.danastreet.com/quote/msft:danalInfo:host=www.xyz.com`

and a script instruction on the page includes:

```
document.location = "/quote/ibm";
```

then the rewritten script instruction would look like:

```
document.location = set_URL("Host=www.xyz.com", "/quote/ibm");
```

and the return result from the set\_URL function would be:

```
/quote/ibm:danalInfo:host=www.xyz.com
```

which would result in a request from the browser for:

`https://secure.danastreet.com/quote/ibm:danalInfo:host=www.xyz.com`

Alternatively, if the script instruction were instead:

```
document.location = "https://www.abc.com/info/msft";
```

then the rewritten script instruction would look like:

```
document.location = set_URL("Host=www.xyz.com",  
5 "https://www.abc.com/info/msft");
```

and the return result from the set\_URL function would be:

```
https://secure.danastreet.com/info/msft:danaInfo:host=www.abc.com
```

Note that, in this case, the first argument to the set\_URL function is not  
needed because the second argument is a full URL and contains all of the  
10 information necessary to construct the final URL.

It should be noted that there are many functions or properties that  
when written to can cause the browser to fetch a URL. Some examples  
include:

```
window.open('url', ...)  
15 form.action = 'url';  
document.location = 'url';  
document.location.replace('url');  
image.src = 'url';
```

20 A fourth example with respect to the string substitution and operation  
1222 is as follows. The string substitution operates to modify properties that  
return a URL. Here, script instructions that read from a property that return a  
URL are replaced with a constant string. In this example, if the script includes

```
var url = document.location;
```

25 such would be replaced by:

```
var url = "http://www.yahoo.com/foo.html";
```

This operation serves to ensure that any script examining its environment would not be confused by the fact that the actual URL of the page is different from what it expects. Note that there is more than one property that may need to be modified. Some examples of properties that can be so modified include:

document.location (returns full URL)

document.domain (returns just the hostname part of URL)

Although the above-described embodiments refer to the use of a single intermediary server within an information retrieval system, it should be recognized that the information retrieval system can also include a plurality of intermediary servers. The various intermediary servers can individually receive requests from client machines and forward them to the appropriate remote servers and return a response back through the intermediary server to the client machine. By having multiple servers, not only can additional processing power be obtained, but also load balancing and localization issues can be addressed. Load balancing can also be facilitated by more particularized secure request processing. For example, for popular (high traffic) websites such as trading.etrade.com, the secure hostname could be specified as "trading.etrade.com.danastreet.com" for the hostname modification approach to provide redirection to the intermediary server. For secure communications, authentication certificates are obtained for the hostnames so that a browser attempting such a secure connection does not alert the user of its absence. In this example then, a certificate for "trading.etrade.com.danastreet.com" would be obtained from a certification authority.

The various aspect of the invention described above can be used alone or in various combinations.

The invention is preferably implemented in software, but can be implemented in hardware or a combination of hardware and software. The

invention can also be embodied as computer readable code on a computer readable medium. The computer readable medium is any data storage device that can store data which can be thereafter be read by a computer system. Examples of the computer readable medium include read-only  
5 memory, random-access memory, CD-ROMs, magnetic tape, optical data storage devices, carrier waves. The computer readable medium can also be distributed over a network coupled computer systems so that the computer readable code is stored and executed in a distributed fashion.

The advantages of the invention are numerous. Different  
10 embodiments or implementations may yield one or more of the following advantages. One advantage of the invention is that script portions of markup language documents can be modified such that even markup language documents using scripts can be properly redirected to access resources on remote servers by way of an intermediary server is facilitated. Another  
15 advantage of the invention is that both secure and unsecure requests can be handled. Still another advantage of the invention is that Universal Resource Locators (URLs), even those within scripts, of markup language documents can be modified.

The many features and advantages of the present invention are  
20 apparent from the written description and, thus, it is intended by the appended claims to cover all such features and advantages of the invention. Further, since numerous modifications and changes will readily occur to those skilled in the art, it is not desired to limit the invention to the exact construction and operation as illustrated and described. Hence, all suitable modifications  
25 and equivalents may be resorted to as falling within the scope of the invention.

*What is claimed is:*



## CLAIMS

1. A method for inserting a toolbar into a webpage at a server machine,  
5 said method comprising:
- receiving a webpage at the server machine to be delivered to a client machine;
- inserting the toolbar into the webpage at the server machine, the toolbar including at least one link to a resource and an executable script; and  
10 delivering the webpage to the client machine.
2. A method as recited in claim 1, wherein the at least one link is a graphical link.
3. A method as recited in claim 1, wherein the executable script is an  
15 activation script.
4. A method as recited in claim 3, wherein the activation script determines whether the toolbar is displayed in the webpage when the webpage is  
20 displayed at the client machine.
5. A method as recited in claim 3, wherein the webpage has at least one frame,
- wherein an instance of the toolbar is inserted into each frame of the  
25 webpage at the server machine.

6. A method as recited in claim 5, wherein, for each of the at least one instances of the toolbar, the activation script determines whether the toolbar is displayed in the at least one frame of the webpage when the webpage is displayed at the client machine.

5

7. A method as recited in claim 5, wherein, for each of the at least one instances of the toolbar, the activation script determines that the toolbar is to be displayed when the webpage is displayed at the client machine when both the activation script is within one of the at least one frame of the webpage and the size of the one of the at least one frame is greater than a threshold size.

10

8. A method as recited in claim 1, wherein the toolbar is a HTML toolbar, and wherein the executable script is provided in a script programming language.

15

9. A method as recited in claim 1, wherein the resource is a remote third-party resource.

20

10. A method as recited in claim 1, wherein the toolbar includes a plurality of links to different resources.

11. A method as recited in claim 1, wherein said method further comprising:

25

executing the executable script on the client machine to determine whether the toolbar should be displayed when displaying the webpage on the client machine.

12. A method for modifying a markup language page to include a dynamically determinable toolbar provided therein, said dynamically determinable toolbar operates when said markup language page is being displayed to dynamically determine whether said dynamically determinable toolbar should be displayed as part of said markup language page.

5

13. A method as recited in claim 12, wherein said dynamically determinable toolbar includes at least one link to a resource and an executable script.

10

14. A method as recited in claim 12, wherein the markup language page includes a least one frame,

wherein said dynamically determinable toolbar is provided within the at least one frame, and

15

wherein said dynamically determinable toolbar operates to display said dynamically determinable toolbar within the at least one frame when the size of the at least one frame is greater than a predetermined size and operates to not display said dynamically determinable toolbar within the at least one frame when the size of the at least one frame is not greater than the predetermined size.

20

15. A method as recited in claim 14, wherein said dynamically determinable toolbar includes at least one link to a remote resource and an executable script.

25

16. A method as recited in claim 14, wherein said dynamically determinable toolbar is a HTML toolbar, and wherein the executable script is provided in a script programming language.

17. A computer readable medium including at least computer program code for determining whether a toolbar should be display in one or more frames of a webpage, said computer readable medium comprising:

computer program code for determining whether the toolbar is within  
5 the one or more frames of the webpage;

computer program code for determining the size of the one or more frames of the webpage; and

computer program code for displaying the toolbar in each of the one or more frames of the webpage in which the size of the one or more frames exceeds a threshold size and suppressing displaying the toolbar in each of  
10 the one or more frames of the webpage in which the size of the one or more frames does not exceed the threshold size.

18. A computer readable medium as recited in claim 17, wherein the webpage and the toolbar are further provided on said computer readable  
15 medium.

19. A computer readable medium including at least computer program code for inserting a toolbar into a webpage at a server machine, said  
20 computer readable medium comprising:

computer program code for receiving a webpage at the server machine to be delivered to a client machine;

computer program code for inserting the toolbar into the webpage at the server machine, the toolbar including at least one link to a resource and  
25 an executable script; and

computer program code for delivering the webpage to the client machine.

20. A computer readable medium as recited in claim 19, wherein the executable script determines whether the toolbar is displayed in the webpage when the webpage is displayed at the client machine.

- 5 21. A dynamically determinable toolbar, said dynamically determinable toolbar being provided in a markup language page, and said dynamically determinable toolbar operating, when the markup language page is being displayed, to dynamically determine whether said dynamically determinable toolbar should be displayed as part of the markup language page.

10

11  
12  
13  
14  
15  
16  
17  
18  
19  
20  
21  
22  
23  
24  
25  
26  
27  
28  
29  
30  
31  
32  
33  
34  
35  
36  
37  
38  
39  
40  
41  
42  
43  
44  
45  
46  
47  
48  
49  
50  
51  
52  
53  
54  
55  
56  
57  
58  
59  
60  
61  
62  
63  
64  
65  
66  
67  
68  
69  
70  
71  
72  
73  
74  
75  
76  
77  
78  
79  
80  
81  
82  
83  
84  
85  
86  
87  
88  
89  
90  
91  
92  
93  
94  
95  
96  
97  
98  
99  
100  
101  
102  
103  
104  
105  
106  
107  
108  
109  
110  
111  
112  
113  
114  
115  
116  
117  
118  
119  
120  
121  
122  
123  
124  
125  
126  
127  
128  
129  
130  
131  
132  
133  
134  
135  
136  
137  
138  
139  
140  
141  
142  
143  
144  
145  
146  
147  
148  
149  
150  
151  
152  
153  
154  
155  
156  
157  
158  
159  
160  
161  
162  
163  
164  
165  
166  
167  
168  
169  
170  
171  
172  
173  
174  
175  
176  
177  
178  
179  
180  
181  
182  
183  
184  
185  
186  
187  
188  
189  
190  
191  
192  
193  
194  
195  
196  
197  
198  
199  
200  
201  
202  
203  
204  
205  
206  
207  
208  
209  
210  
211  
212  
213  
214  
215  
216  
217  
218  
219  
220  
221  
222  
223  
224  
225  
226  
227  
228  
229  
230  
231  
232  
233  
234  
235  
236  
237  
238  
239  
240  
241  
242  
243  
244  
245  
246  
247  
248  
249  
250  
251  
252  
253  
254  
255  
256  
257  
258  
259  
260  
261  
262  
263  
264  
265  
266  
267  
268  
269  
270  
271  
272  
273  
274  
275  
276  
277  
278  
279  
280  
281  
282  
283  
284  
285  
286  
287  
288  
289  
290  
291  
292  
293  
294  
295  
296  
297  
298  
299  
300  
301  
302  
303  
304  
305  
306  
307  
308  
309  
310  
311  
312  
313  
314  
315  
316  
317  
318  
319  
320  
321  
322  
323  
324  
325  
326  
327  
328  
329  
330  
331  
332  
333  
334  
335  
336  
337  
338  
339  
340  
341  
342  
343  
344  
345  
346  
347  
348  
349  
350  
351  
352  
353  
354  
355  
356  
357  
358  
359  
360  
361  
362  
363  
364  
365  
366  
367  
368  
369  
370  
371  
372  
373  
374  
375  
376  
377  
378  
379  
380  
381  
382  
383  
384  
385  
386  
387  
388  
389  
390  
391  
392  
393  
394  
395  
396  
397  
398  
399  
400  
401  
402  
403  
404  
405  
406  
407  
408  
409  
410  
411  
412  
413  
414  
415  
416  
417  
418  
419  
420  
421  
422  
423  
424  
425  
426  
427  
428  
429  
430  
431  
432  
433  
434  
435  
436  
437  
438  
439  
440  
441  
442  
443  
444  
445  
446  
447  
448  
449  
450  
451  
452  
453  
454  
455  
456  
457  
458  
459  
460  
461  
462  
463  
464  
465  
466  
467  
468  
469  
470  
471  
472  
473  
474  
475  
476  
477  
478  
479  
480  
481  
482  
483  
484  
485  
486  
487  
488  
489  
490  
491  
492  
493  
494  
495  
496  
497  
498  
499  
500  
501  
502  
503  
504  
505  
506  
507  
508  
509  
510  
511  
512  
513  
514  
515  
516  
517  
518  
519  
520  
521  
522  
523  
524  
525  
526  
527  
528  
529  
530  
531  
532  
533  
534  
535  
536  
537  
538  
539  
540  
541  
542  
543  
544  
545  
546  
547  
548  
549  
550  
551  
552  
553  
554  
555  
556  
557  
558  
559  
560  
561  
562  
563  
564  
565  
566  
567  
568  
569  
570  
571  
572  
573  
574  
575  
576  
577  
578  
579  
580  
581  
582  
583  
584  
585  
586  
587  
588  
589  
590  
591  
592  
593  
594  
595  
596  
597  
598  
599  
600  
601  
602  
603  
604  
605  
606  
607  
608  
609  
610  
611  
612  
613  
614  
615  
616  
617  
618  
619  
620  
621  
622  
623  
624  
625  
626  
627  
628  
629  
630  
631  
632  
633  
634  
635  
636  
637  
638  
639  
640  
641  
642  
643  
644  
645  
646  
647  
648  
649  
650  
651  
652  
653  
654  
655  
656  
657  
658  
659  
660  
661  
662  
663  
664  
665  
666  
667  
668  
669  
670  
671  
672  
673  
674  
675  
676  
677  
678  
679  
680  
681  
682  
683  
684  
685  
686  
687  
688  
689  
690  
691  
692  
693  
694  
695  
696  
697  
698  
699  
700  
701  
702  
703  
704  
705  
706  
707  
708  
709  
710  
711  
712  
713  
714  
715  
716  
717  
718  
719  
720  
721  
722  
723  
724  
725  
726  
727  
728  
729  
730  
731  
732  
733  
734  
735  
736  
737  
738  
739  
740  
741  
742  
743  
744  
745  
746  
747  
748  
749  
750  
751  
752  
753  
754  
755  
756  
757  
758  
759  
760  
761  
762  
763  
764  
765  
766  
767  
768  
769  
770  
771  
772  
773  
774  
775  
776  
777  
778  
779  
780  
781  
782  
783  
784  
785  
786  
787  
788  
789  
790  
791  
792  
793  
794  
795  
796  
797  
798  
799  
800  
801  
802  
803  
804  
805  
806  
807  
808  
809  
810  
811  
812  
813  
814  
815  
816  
817  
818  
819  
820  
821  
822  
823  
824  
825  
826  
827  
828  
829  
830  
831  
832  
833  
834  
835  
836  
837  
838  
839  
840  
841  
842  
843  
844  
845  
846  
847  
848  
849  
850  
851  
852  
853  
854  
855  
856  
857  
858  
859  
860  
861  
862  
863  
864  
865  
866  
867  
868  
869  
870  
871  
872  
873  
874  
875  
876  
877  
878  
879  
880  
881  
882  
883  
884  
885  
886  
887  
888  
889  
890  
891  
892  
893  
894  
895  
896  
897  
898  
899  
900  
901  
902  
903  
904  
905  
906  
907  
908  
909  
910  
911  
912  
913  
914  
915  
916  
917  
918  
919  
920  
921  
922  
923  
924  
925  
926  
927  
928  
929  
930  
931  
932  
933  
934  
935  
936  
937  
938  
939  
940  
941  
942  
943  
944  
945  
946  
947  
948  
949  
950  
951  
952  
953  
954  
955  
956  
957  
958  
959  
960  
961  
962  
963  
964  
965  
966  
967  
968  
969  
970  
971  
972  
973  
974  
975  
976  
977  
978  
979  
980  
981  
982  
983  
984  
985  
986  
987  
988  
989  
990  
991  
992  
993  
994  
995  
996  
997  
998  
999  
1000  
1001  
1002  
1003  
1004  
1005  
1006  
1007  
1008  
1009  
1010  
1011  
1012  
1013  
1014  
1015  
1016  
1017  
1018  
1019  
1020  
1021  
1022  
1023  
1024  
1025  
1026  
1027  
1028  
1029  
1030  
1031  
1032  
1033  
1034  
1035  
1036  
1037  
1038  
1039  
1040  
1041  
1042  
1043  
1044  
1045  
1046  
1047  
1048  
1049  
1050  
1051  
1052  
1053  
1054  
1055  
1056  
1057  
1058  
1059  
1060  
1061  
1062  
1063  
1064  
1065  
1066  
1067  
1068  
1069  
1070  
1071  
1072  
1073  
1074  
1075  
1076  
1077  
1078  
1079  
1080  
1081  
1082  
1083  
1084  
1085  
1086  
1087  
1088  
1089  
1090  
1091  
1092  
1093  
1094  
1095  
1096  
1097  
1098  
1099  
1100  
1101  
1102  
1103  
1104  
1105  
1106  
1107  
1108  
1109  
1110  
1111  
1112  
1113  
1114  
1115  
1116  
1117  
1118  
1119  
1120  
1121  
1122  
1123  
1124  
1125  
1126  
1127  
1128  
1129  
1130  
1131  
1132  
1133  
1134  
1135  
1136  
1137  
1138  
1139  
1140  
1141  
1142  
1143  
1144  
1145  
1146  
1147  
1148  
1149  
1150  
1151  
1152  
1153  
1154  
1155  
1156  
1157  
1158  
1159  
1160  
1161  
1162  
1163  
1164  
1165  
1166  
1167  
1168  
1169  
1170  
1171  
1172  
1173  
1174  
1175  
1176  
1177  
1178  
1179  
1180  
1181  
1182  
1183  
1184  
1185  
1186  
1187  
1188  
1189  
1190  
1191  
1192  
1193  
1194  
1195  
1196  
1197  
1198  
1199  
1200  
1201  
1202  
1203  
1204  
1205  
1206  
1207  
1208  
1209  
1210  
1211  
1212  
1213  
1214  
1215  
1216  
1217  
1218  
1219  
1220  
1221  
1222  
1223  
1224  
1225  
1226  
1227  
1228  
1229  
1230  
1231  
1232  
1233  
1234  
1235  
1236  
1237  
1238  
1239  
1240  
1241  
1242  
1243  
1244  
1245  
1246  
1247  
1248  
1249  
1250  
1251  
1252  
1253  
1254  
1255  
1256  
1257  
1258  
1259  
1260  
1261  
1262  
1263  
1264  
1265  
1266  
1267  
1268  
1269  
1270  
1271  
1272  
1273  
1274  
1275  
1276  
1277  
1278  
1279  
1280  
1281  
1282  
1283  
1284  
1285  
1286  
1287  
1288  
1289  
1290  
1291  
1292  
1293  
1294  
1295  
1296  
1297  
1298  
1299  
1300  
1301  
1302  
1303  
1304  
1305  
1306  
1307  
1308  
1309  
1310  
1311  
1312  
1313  
1314  
1315  
1316  
1317  
1318  
1319  
1320  
1321  
1322  
1323  
1324  
1325  
1326  
1327  
1328  
1329  
1330  
1331  
1332  
1333  
1334  
1335  
1336  
1337  
1338  
1339  
1340  
1341  
1342  
1343  
1344  
1345  
1346  
1347  
1348  
1349  
1350  
1351  
1352  
1353  
1354  
1355  
1356  
1357  
1358  
1359  
1360  
1361  
1362  
1363  
1364  
1365  
1366  
1367  
1368  
1369  
1370  
1371  
1372  
1373  
1374  
1375  
1376  
1377  
1378  
1379  
1380  
1381  
1382  
1383  
1384  
1385  
1386  
1387  
1388  
1389  
1390  
1391  
1392  
1393  
1394  
1395  
1396  
1397  
1398  
1399  
1400  
1401  
1402  
1403  
1404  
1405  
1406  
1407  
1408  
1409  
1410  
1411  
1412  
1413  
1414  
1415  
1416  
1417  
1418  
1419  
1420  
1421  
1422  
1423  
1424  
1425  
1426  
1427  
1428  
1429  
1430  
1431  
1432  
1433  
1434  
1435  
1436  
1437  
1438  
1439  
1440  
1441  
1442  
1443  
1444  
1445  
1446  
1447  
1448  
1449  
1450  
1451  
1452  
1453  
1454  
1455  
1456  
1457  
1458  
1459  
1460  
1461  
1462  
1463  
1464  
1465  
1466  
1467  
1468  
1469  
1470  
1471  
1472  
1473  
1474  
1475  
1476  
1477  
1478  
1479  
1480  
1481  
1482  
1483  
1484  
1485  
1486  
1487  
1488  
1489  
1490  
1491  
1492  
1493  
1494  
1495  
1496  
1497  
1498  
1499  
1500  
1501  
1502  
1503  
1504  
1505  
1506  
1507  
1508  
1509  
1510  
1511  
1512  
1513  
1514  
1515  
1516  
1517  
1518  
1519  
1520  
1521  
1522  
1523  
1524  
1525  
1526  
1527  
1528  
1529  
1530  
1531  
1532  
1533  
1534  
1535  
1536  
1537  
1538  
1539  
1540  
1541  
1542  
1543  
1544  
1545  
1546  
1547  
1548  
1549  
1550  
1551  
1552  
1553  
1554  
1555  
1556  
1557  
1558  
1559  
1560  
1561  
1562  
1563  
1564  
1565  
1566  
1567  
1568  
1569  
1570  
1571  
1572  
1573  
1574  
1575  
1576  
1577  
1578  
1579  
1580  
1581  
1582  
1583  
1584  
1585  
1586  
1587  
1588  
1589  
1590  
1591  
1592  
1593  
1594  
1595  
1596  
1597  
1598  
1599  
1600  
1601  
1602  
1603  
1604  
1605  
1606  
1607  
1608  
1609  
1610  
1611  
1612  
1613  
1614  
1615  
1616  
1617  
1618  
1619  
1620  
1621  
1622  
1623  
1624  
1625  
1626  
1627  
1628  
1629  
1630  
1631  
1632  
1633  
1634  
1635  
1636  
1637  
1638  
1639  
1640  
1641  
1642  
1643  
1644  
1645  
1646  
1647  
1648  
1649  
1650  
1651  
1652  
1653  
1654  
1655  
1656  
1657  
1658  
1659  
1660  
1661  
1662  
1663  
1664  
1665  
1666  
1667  
1668  
1669  
1670  
1671  
1672  
1673  
1674  
1675  
1676  
1677  
1678  
1679  
1680  
1681  
1682  
1683  
1684  
1685  
1686  
1687  
1688  
1689  
1690  
1691  
1692  
1693  
1694  
1695  
1696  
1697  
1698  
1699  
1700  
1701  
1702  
1703  
1704  
1705  
1706  
1707  
1708  
1709  
1710  
1711  
1712  
1713  
1714  
1715  
1716  
1717  
1718  
1719  
1720  
1721  
1722  
1723  
1724  
1725  
1726  
1727  
1728  
1729  
1730  
1731  
1732  
1733  
1734  
1735  
1736  
1737  
1738  
1739  
1740  
1741  
1742  
1743  
1744  
1745  
1746  
1747  
1748  
1749  
1750  
1751  
1752  
1753  
1754  
1755  
1756  
1757  
1758  
1759  
1760  
1761  
1762  
1763  
1764  
1765  
1766  
1767  
1768  
1769  
1770  
1771  
1772  
1773  
1774  
1775  
1776  
1777  
1778  
1779  
1780  
1781  
1782  
1783  
1784  
1785  
1786  
1787  
1788  
1789  
1790  
1791  
1792  
1793  
1794  
1795  
1796  
1797  
1798  
1799  
1800  
1801  
1802  
1803  
1804  
1805  
1806  
1807  
1808  
1809  
1810  
1811  
1812  
1813  
1814  
1815  
1816  
1817  
1818  
1819  
1820  
1821  
1822  
1823  
1824  
1825  
1826  
1827  
1828  
1829  
1830  
1831  
1832  
1833  
1834  
1835  
1836  
1837  
1838  
1839  
1840  
1841  
1842  
1843  
1844  
1845  
1846  
1847  
1848  
1849  
1850  
1851  
1852  
1853  
1854  
1855  
1856  
1857  
1858  
1859  
1860  
1861  
1862  
1863  
1864  
1865  
1866  
1867  
1868  
1869  
1870  
1871  
1872  
1873  
1874  
1875  
1876  
1877  
1878  
1879  
1880  
1881  
1882  
1883  
1884  
1885  
1886  
1887  
1888  
1889  
1890  
1891  
1892  
1893  
1894  
1895  
1896  
1897  
1898  
1899  
1900  
1901  
1902  
1903  
1904  
1905  
1906  
1907  
1908  
1909  
1910  
1911  
1912  
1913  
1914  
1915  
1916  
1917  
1918  
1919  
1920  
1921  
1922  
1923  
1924  
1925  
1926  
1927  
1928  
1929  
1930  
1931  
1932  
1933  
1934  
1935  
1936  
1937  
1938  
1939  
1940  
1941  
1942  
1943  
1944  
1945  
1946  
1947  
1948  
1949  
1950  
1951  
1952  
1953  
1954  
1955  
1956  
1957  
1958  
1959  
1960  
1961  
1962  
1963  
1964  
1965  
1966  
1967  
1968  
1969  
1970  
1971  
1972  
1973  
1974  
1975  
1976  
1977  
1978  
1979  
1980  
1981  
1982  
1983  
1984  
1985  
1986  
1987  
1988  
1989  
1990  
1991  
1992  
1993  
1994  
1995  
1996  
1997  
1998  
1999  
2000  
2001  
2002  
2003  
2004  
2005  
2006  
2007  
2008  
2009  
2010  
2011  
2012  
2013  
2014  
2015  
2016  
2017  
2018  
2019  
2020  
2021  
2022  
2023  
2024  
2025  
2026  
2027  
2028  
2029  
2030  
2031  
2032  
2033  
2034  
2035  
2036  
2037  
2038  
2039  
2040  
2041  
2042  
2043  
2044  
2045  
2046  
2047  
2048  
2049  
2050  
2051  
2052  
2053  
2054  
2055  
2056  
2057  
2058  
2059  
2060  
2061  
2062  
2063  
2064  
2065  
2066  
2067  
2068  
2069  
2070  
2071  
2072  
2073  
2074  
2075  
2076  
2077  
2078  
2079  
2080  
2081  
2082  
2083  
2084  
2085  
2086  
2087  
2088  
2089  
2090  
2091  
2092  
2093  
2094  
2095  
2096  
2097  
2098  
2099  
2100  
2101  
2102  
2103  
2104  
2105  
2106  
2107  
2108  
2109  
2110  
2111  
2112  
2113  
2114  
2115  
2116  
2117  
2118  
2119  
2120  
2121  
2122  
2123  
2124  
2125  
2126  
2127  
2128  
2129  
2130  
2131  
2132  
2133  
2134  
2135  
2136  
2137  
2138  
2139  
2140  
2141  
2142  
2143  
2144  
2145  
2146  
2147  
2148  
2149  
2150  
2151  
2152  
2153  
2154  
2155  
2156  
2157  
2158  
2159  
2160  
2161  
2162  
2163  
2164  
2165  
2166  
2167  
2168  
2169  
2170  
2171  
2172  
2173  
2174  
2175  
2176  
2177  
2178  
2179  
2180  
2181  
2182  
2183  
2184  
2185  
2186  
2187  
2188  
2189  
2190  
2191  
2192  
2193  
2194  
2195  
2196  
2197  
2198  
2199  
2200  
2201  
2202  
2203  
2204  
2205  
2206  
2207  
2208  
2209  
2210  
2211  
2212  
2213  
2214  
2215  
2216

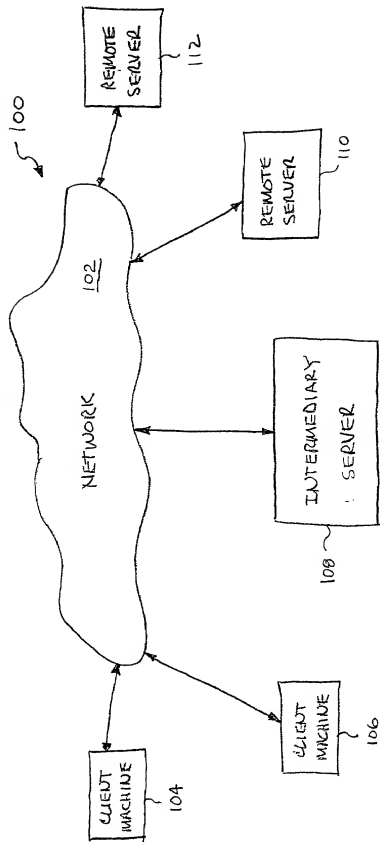
## DYNAMIC TOOLBAR FOR MARKUP LANGUAGE DOCUMENT

5

### ABSTRACT OF THE DISCLOSURE

A toolbar that is provided or inserted in a markup language document so as to facilitate features or functionality provided by a server is disclosed. The toolbar is able to determine whether the toolbar should be displayed as part of the markup language page being displayed. In one embodiment the server is an intermediary server.

10



4/5

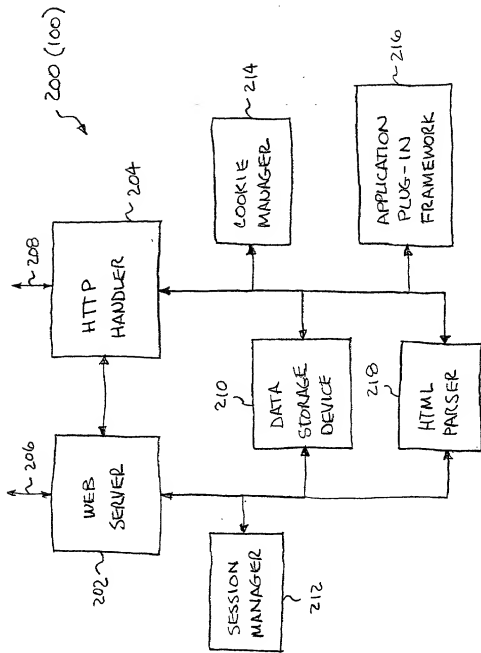
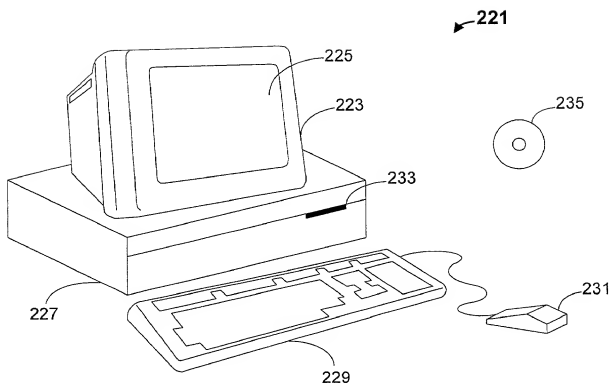
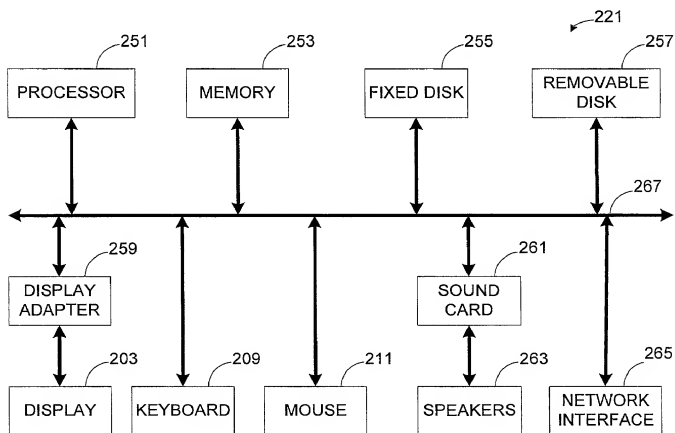


FIG. 2A





**FIG. 2B**



**FIG. 2C**

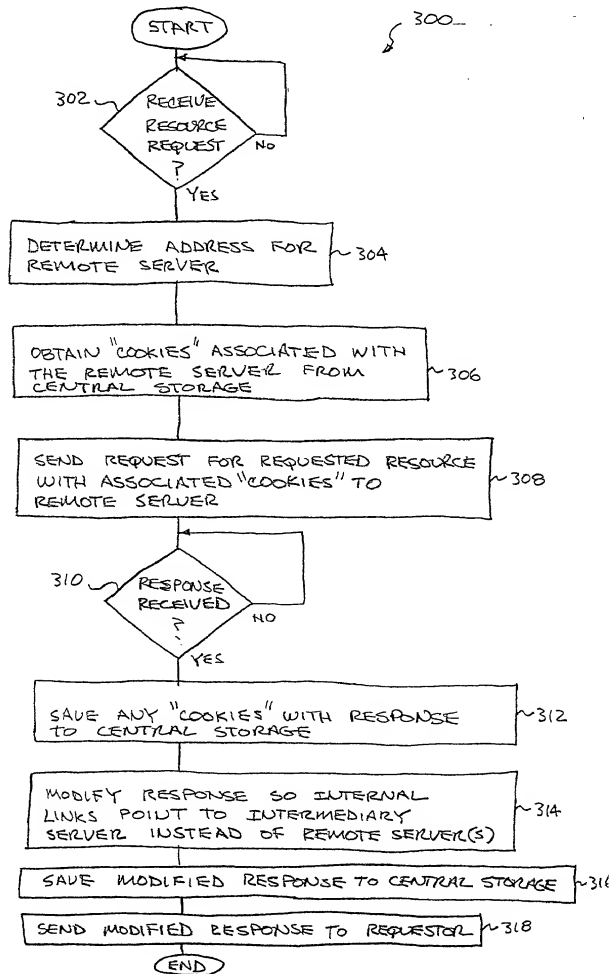
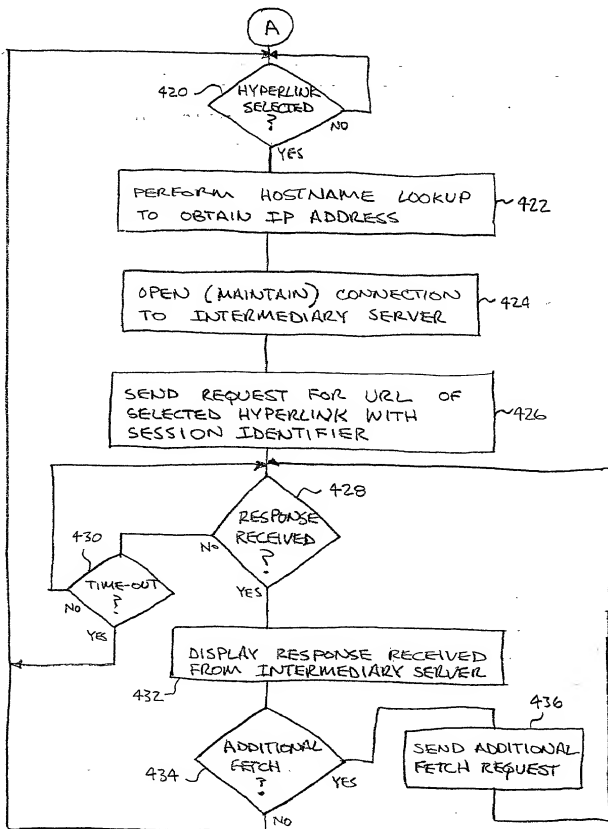


FIG.3





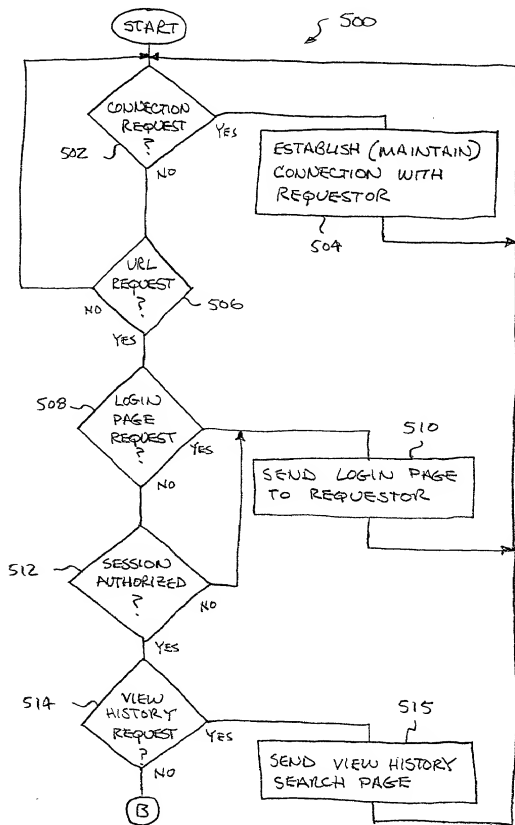


FIG. 5A

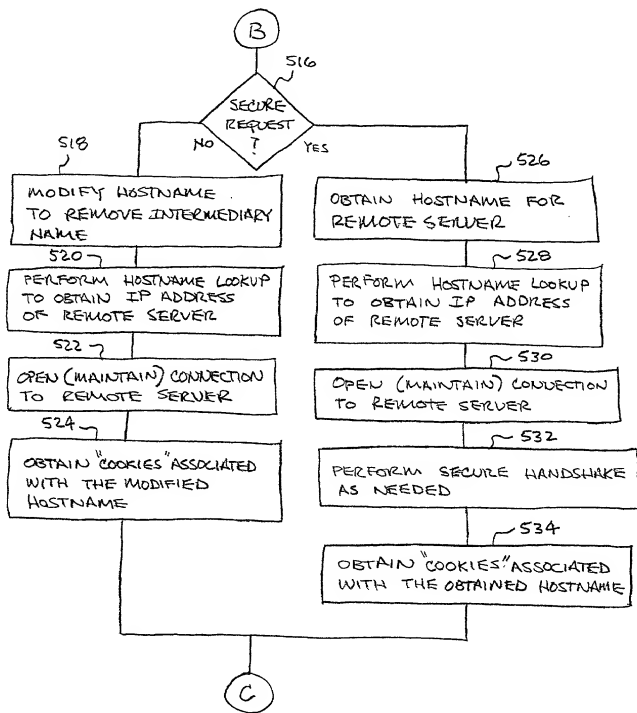


FIG. 5B

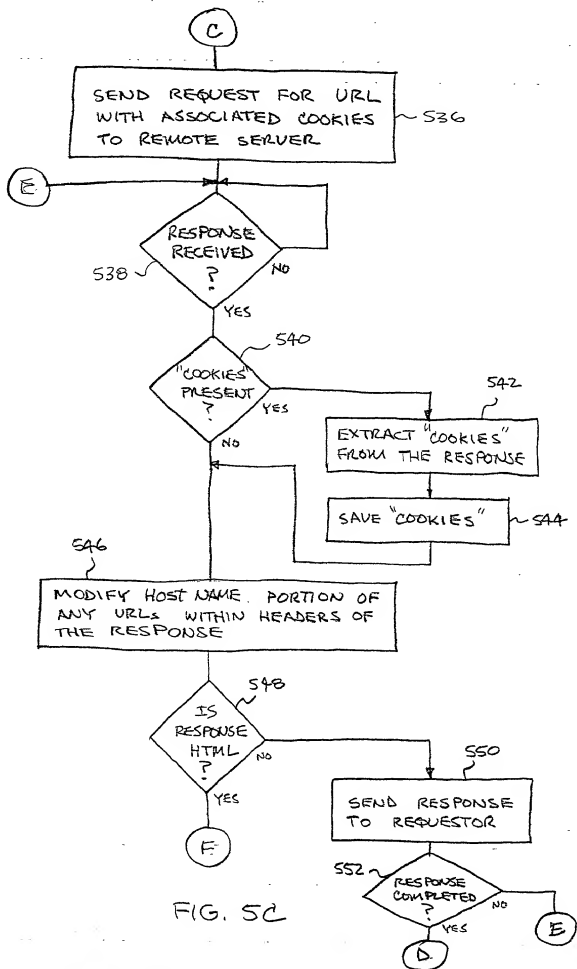
[illegible]

FIG. 5C

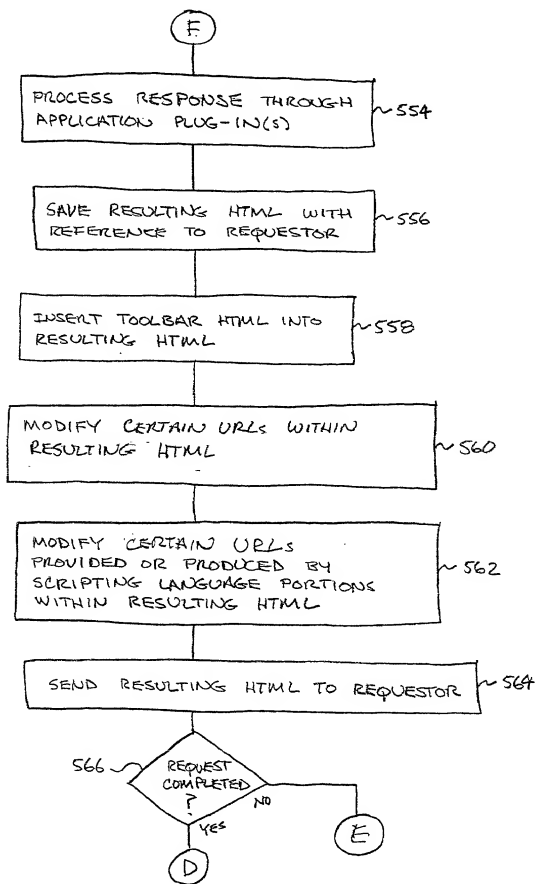


FIG. 5D



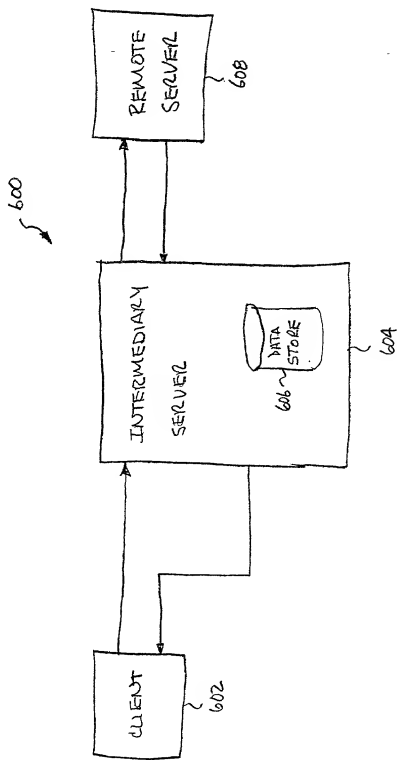


FIG. 6

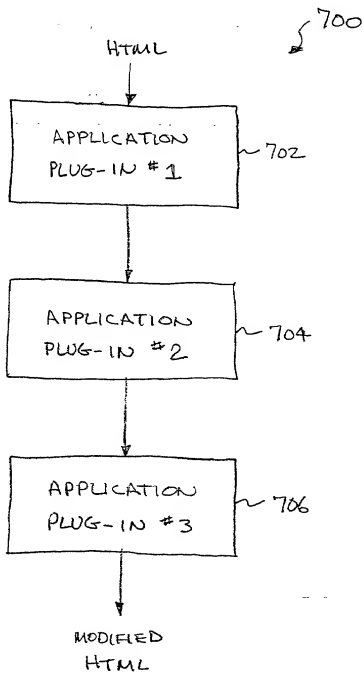


FIG. 7

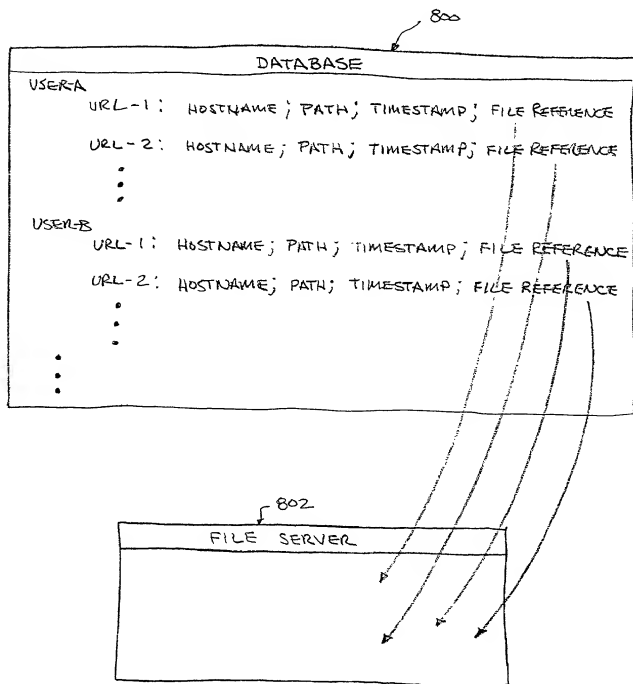


FIG. 8

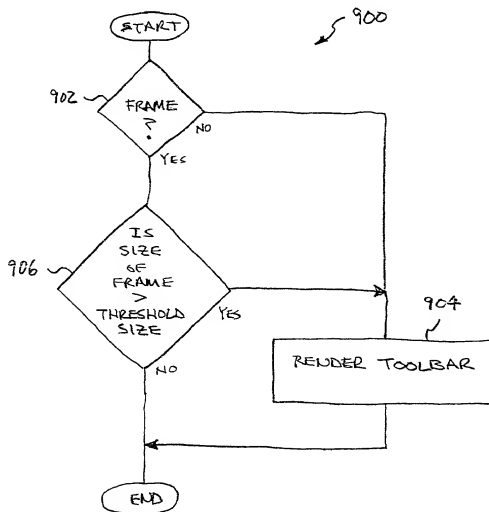


FIG. 9A

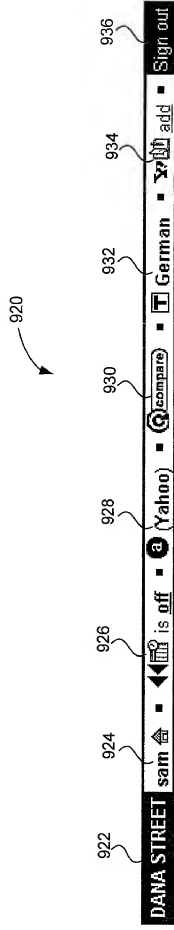
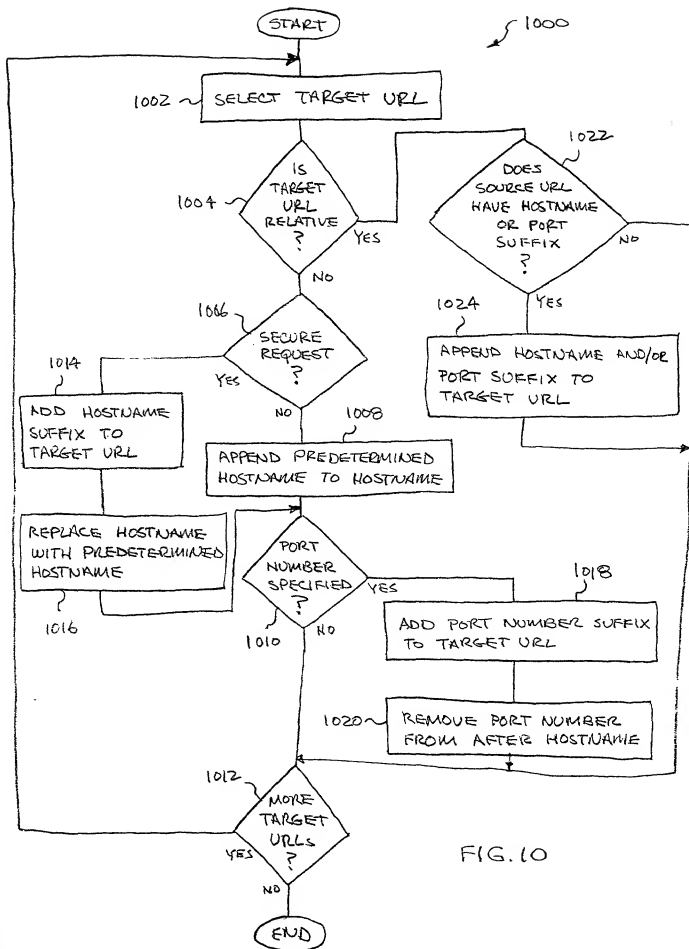


FIG. 9B



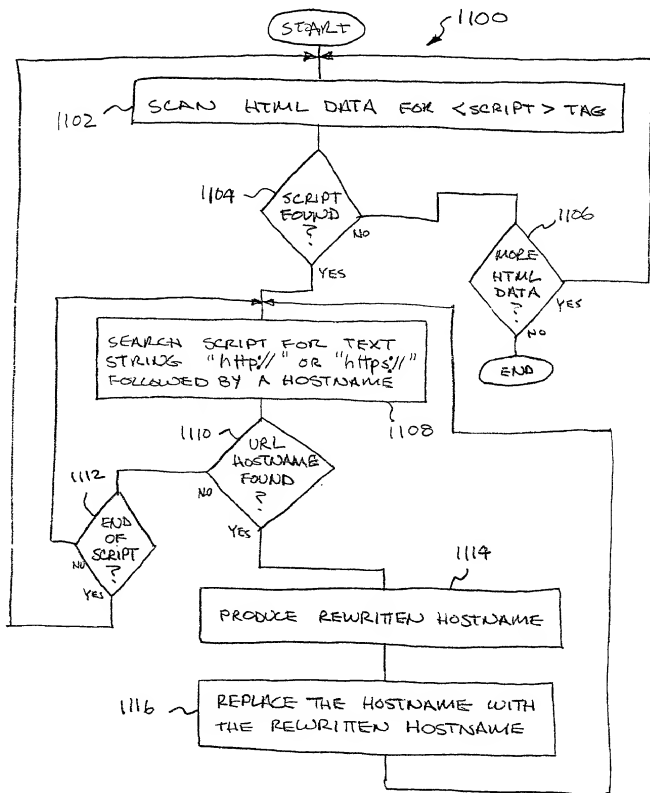


FIG. 11

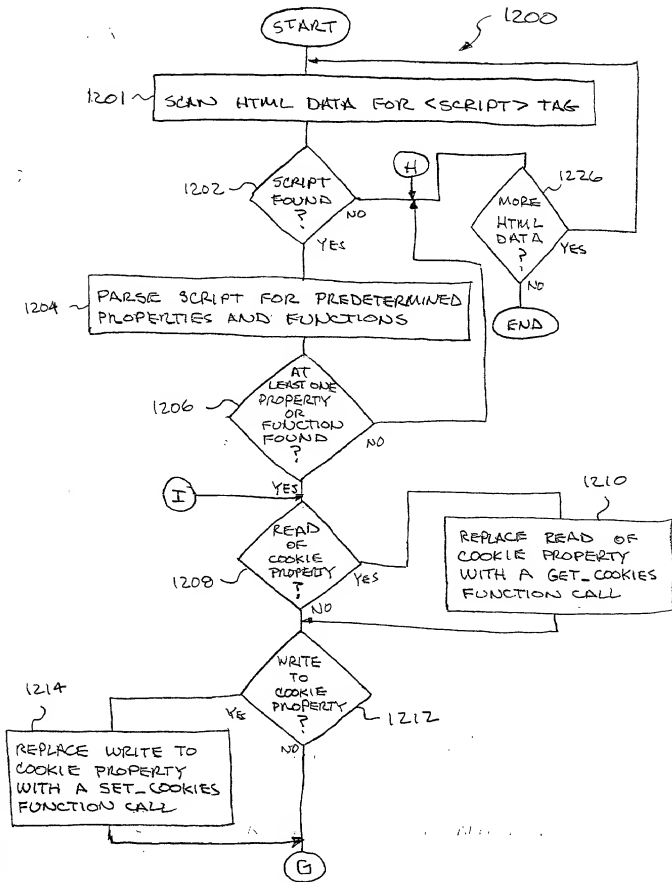


FIG. 12A



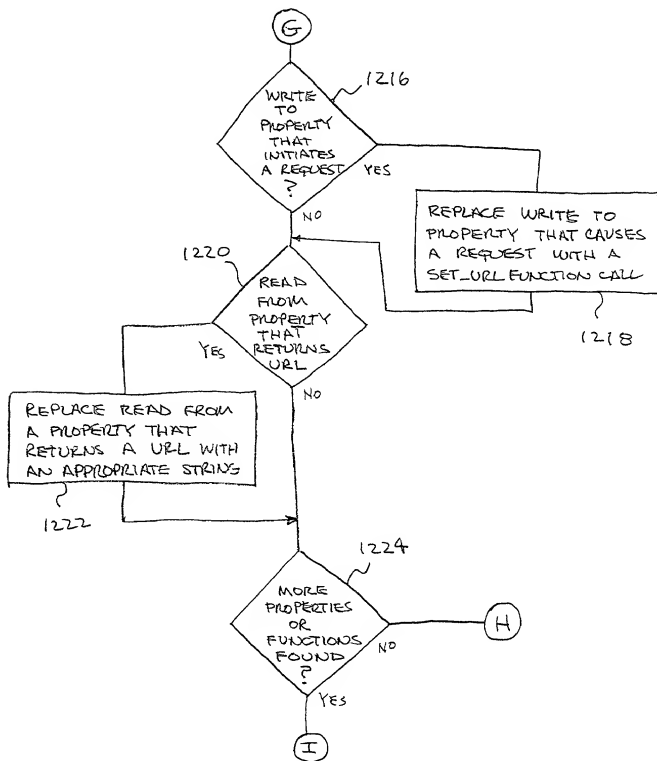


FIG. 12B

# DECLARATION AND POWER OF ATTORNEY FOR ORIGINAL U.S. PATENT APPLICATION

Attorney's Docket No. DANAP004

As a below-named inventor, I hereby declare that:

My residence, post office address and citizenship are as stated below next to my name.

I believe that I am the original, first and sole inventor (if only one name is listed below) or an original, first and joint inventor (if plural names are listed below) of the subject matter which is claimed and for which a patent is sought on the invention entitled: DYNAMIC TOOLBAR FOR MARKUP LANGUAGE DOCUMENT, the specification of which,

(check one)

1. ☒ is attached hereto.

2. ☐ was filed on \_\_\_\_\_ as  
U.S. Application No. \_\_\_\_\_  
and was amended on \_\_\_\_\_;

3. ☐ was filed on \_\_\_\_\_ as  
International PCT Application No. \_\_\_\_\_  
and was amended on \_\_\_\_\_;

I hereby state that I have reviewed and understand the contents of the above-identified specification, including the claims, as amended by any amendment referred to above.

I acknowledge the duty to disclose information which is material to the examination of this application in accordance with Title 37, CFR § 1.56.

## Prior Foreign Application(s)

I hereby claim foreign priority benefits under Title 35, United States code, § 119(a)-(d) or § 365(b) of any foreign application(s) for patent or inventor's certificate, or § 365(a) of any PCT International application which designated at least one country other than the United States, listed below and have identified below, by checking the box, any foreign application for patent or inventor's certificate, or PCT International application having a filing date before that of the application on which priority is claimed:

			Priority Benefits Claimed?
_____ (Application No.)	_____ (Country)	_____ (Filing Date)	Yes ___ No ___
_____ (Application No.)	_____ (Country)	_____ (Filing Date)	Yes ___ No ___

## Provisional Application(s)

I hereby claim the benefit under 35 U.S.C. § 119(e) of any United States provisional application(s) listed below:

60/235,513 (Application No.)	September 26, 2000 (Filing Date)
_____ (Application No.)	_____ (Filing Date)

**Prior U.S. Application(s)**

I hereby claim the benefit under Title 35, United States Code, § 120 of any United States application(s), or § 365(c) of any PCT International application designating the United States, listed below and, insofar as the subject matter of each of the claims of this application is not disclosed in the prior United States or PCT International application in the manner provided by the first paragraph of Title 35, United States Code, § 112, I acknowledge the duty to disclose information which is material to patentability as defined in Title 37, Code of Federal Regulations, § 1.56 which became available between the filing date of the prior application and the national or PCT international filing date of this application:

_____ (Application No.)	_____ (Filing Date)	_____ (Status - patented, pending, abandoned)
----------------------------	------------------------	--

_____ (Application No.)	_____ (Filing Date)	_____ (Status - patented, pending, abandoned)
----------------------------	------------------------	--

**Power of Attorney**

And I hereby appoint the law firm of **Beyer Weaver & Thomas, LLP** and all practitioners who are associated with the Customer Number 022434 as my principal attorneys to prosecute this application and to transact all business in the Patent and Trademark Office connected therewith.

**Direct Correspondence To:**

**Customer Number: 022434**  
**BEYER WEAVER & THOMAS, LLP**  
P.O. Box 778  
Berkeley, CA 94704-0778



**Direct Telephone Calls To:**

**C. Douglass Thomas at telephone number (650) 961-8300**

I hereby declare that all statements made herein of my own knowledge are true and that all statements made on information and belief are believed to be true; and further that these statements were made with the knowledge that willful false statements and the like so made are punishable by fine or imprisonment, or both, under section 1001 of Title 18 of the United States Code, and that such willful false statements may jeopardize the validity of the application or any patent issuing thereon.

**Typewritten Full Name of**

**Sole or First Inventor:** Sampath Srinivas **Citizenship:** USA

**Inventor's signature:** *AS* **Date of Signature:** 03 NOV 2000

**Residence:** (City) Santa Clara **(State/Country)** CA/USA

**Post Office Address:** 3443 Georgetown Place, Santa Clara, CA 95051

**Second Inventor:** Theron Tock **Citizenship:** USA

**Inventor's signature:** *Theron Tock* **Date of Signature:** 03 NOV 2000

**Residence:** (City) Mountain View **(State/Country)** CA/USA

**Post Office Address:** 620 Palo Alto Avenue, Mountain View, CA 94041